



User Guide

AWS Elemental MediaPackage v2



AWS Elemental MediaPackage v2: User Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is AWS Elemental MediaPackage?	1
Are you a first-time user of MediaPackage?	2
Concepts and terminology	2
Supported inputs and outputs	4
Supported input types	4
Supported input codecs	5
Supported output codecs	5
How MediaPackage works	7
General live processing flow	8
Live input redundancy processing flow	9
Supported features of AWS Elemental MediaPackage	11
Related services	13
Accessing MediaPackage	14
Pricing for MediaPackage	14
Regions for MediaPackage	15
AWS opt-in Regions	15
Setting up MediaPackage	17
Sign up for an AWS account	17
Set up additional IAM permissions	17
Create a role in the IAM console	18
Assume the role from the IAM console or AWS CLI	20
Resource Groups—tagging	20
Allowing MediaPackage to access other AWS services	20
Step 1: Create a policy	21
Step 2: Create a role	24
Step 3: Modify the trust relationship	25
Download tools	27
IPv6 support	28
IPv6 endpoints	28
Using IPv6 endpoints	28
Getting started	30
Prerequisites	30
Step 1: Access MediaPackage	30
Step 2: Create a channel group	30

Step 3: Create a channel	31
Step 4: Create an endpoint	31
Step 5: Clean up	32
Access control	33
Creating new resources	33
Sharing resources	34
Protecting data	34
Delivering live content	36
Working with channel groups	36
Creating a channel group	37
Viewing channel group details	38
Editing a channel group	39
Deleting a channel group	39
Working with channels	40
Creating a channel	41
Viewing channel details	44
Editing a channel	45
Resetting channel history	45
Deleting a channel	46
Working with endpoints	46
Creating an origin endpoint	47
Viewing an origin endpoint	70
Editing an endpoint	71
Resetting an endpoint	71
Deleting an endpoint	72
Previewing a manifest	72
Delivering VOD content	74
Creating live-to-VOD assets	75
Requirements	75
What is a harvest job?	76
How live-to-VOD works	76
Live-to-VOD HarvestJob	76
Live-to-VOD HarvestObject	77
Creating a harvest job	78
Viewing harvest jobs	79
Canceling a harvest job	80

MediaPackage features	81
DASH	81
DASH features and capabilities	82
DASH manifest structure	82
DASH period triggers	83
DASH use cases	83
CMAF ingest	83
Choosing output group	84
Enhanced CMAF capabilities	85
Output locking mode	86
A/B watermarking	88
Channel requirements	89
How watermarking affects segment combining	90
WMPaceInfoIngest header	90
Content encryption and DRM	91
Limitations and requirements	91
Container and DRM system support with SPEKE	92
Deploying SPEKE	93
Implementing SPEKE v2.0	94
Content key encryption	103
Key rotation	106
Managing DRM segment metadata	106
Exclude session keys	109
Encryption presets	110
CMSD headers	114
Common CMSD keys	114
AWS custom CMSD keys	117
Cross-Region failover	117
Requirements for cross-Region failover in AWS Elemental MediaPackage	118
Encoder restrictions for cross-Region failover	119
DASH manifest treatments	119
Multi-period DASH	121
DASH manifest compactness	122
DASH audio timeline pattern	124
HLS and LL-HLS	128
HLS and LL-HLS features and capabilities	128

HLS manifest structure	129
Differences between HLS and LL-HLS	129
HLS and LL-HLS use cases	130
Dynamic Multiview	130
MediaPackage multiview channels	131
Layouts	132
Requirements	135
Set up a multiview workflow	136
Request a multiview stream	137
Audio, captions, and source loss	137
Identify a view's audio and captions	138
Ad signaling (SCTE-35)	139
CDN caching	139
Constraints	139
Manifest filtering	140
Manifest filtering query parameters	141
Special conditions for TS and CMAF manifests	153
Manifest filtering examples	154
Manifest filtering error conditions	155
Media quality scores	156
How MQCS works	157
Requirements for using MQCS	158
Metadata passthrough	158
ID3 metadata considerations	159
KLV metadata considerations	159
Microsoft Smooth Streaming	160
When to use MSS	161
Choosing the right streaming protocol for your audience	161
Planning your MSS implementation	163
MSS use cases	164
MSS manifest structure	165
MSS encryption	169
Testing MSS playback	172
Troubleshooting MSS	174
CDN configuration for MSS	184
Rendition groups	186

When to use rendition groups	187
When not to use rendition groups	187
Reset for channels and endpoints	188
SCTE-35 messages	189
How it works	190
SCTE-35 settings	191
HLS EXT-X-DATERANGE ad markers	194
HLS CUE tag ad markers	197
DASH ad markers	199
Time-shifted viewing	204
Step 1: Set the startover window	205
Step 2: Choose time-shifted options	205
Time-shifted query parameters	206
Time delay	212
Start and end parameters	214
Window duration	215
Time-shifted viewing examples	215
Trick-play	217
Using I-frame playlists	217
Using image media playlists	221
Working with CDNs	228
CDN configuration recommendations	229
Honor MediaPackage 'cache-control: max-age' values	229
Include specific query strings in your CDN cache key	229
Response timeout	45
Forwarded HTTP headers	230
Forwarded cookies	230
CDN authorization	230
How it works	230
Setup CDN authorization	232
Rotate CDN secrets	235
Troubleshoot CDN authorization	236
CDN authorization best practices	237
Data plane APIs	238
PutObject	238
GetObject	238

GetHeadObject	238
HarvestObject	239
Security	240
Data protection	240
Implementing DRM	242
Identity and Access Management	242
Audience	242
Authenticating with identities	243
Managing access using policies	244
How AWS Elemental MediaPackage works with IAM	246
Identity-based policy examples	253
Resource-based policy examples	256
AWS managed policies	269
Authenticating Requests	271
Cross-service confused deputy prevention	272
Troubleshooting	273
Learn More	275
Compliance validation	275
Resilience	276
Infrastructure Security	276
Logging and monitoring	277
Monitoring with CloudWatch metrics	277
Live content metrics	279
MediaPackage live dimensions	292
Dynamic Multiview metrics	298
Monitoring with EventBridge	302
MediaPackage V2 events	303
Creating event notifications	306
Logging AWS Elemental MediaPackage API calls with AWS CloudTrail	307
MediaPackage information in CloudTrail	307
Understanding MediaPackage log file entries	308
Access logging	310
Permissions	311
Enable access logging	312
Manage and disable access logging	315
Read access logs	315

Monitoring manifest update time in AWS Elemental MediaPackage	320
HLS manifest example	320
MediaPackage response headers	321
Tagging resources	324
Tag restrictions	324
Managing tags	324
Quotas	326
Soft quotas	326
Hard quotas	327
Document history	329

What is AWS Elemental MediaPackage?

AWS Elemental MediaPackage (MediaPackage) is a just-in-time video packaging and origination service that runs in the AWS Cloud. With MediaPackage, you can deliver highly secure, scalable, and reliable video streams to a wide variety of playback devices and content delivery networks (CDNs).

MediaPackage offers a broadcast-grade viewing experience for viewers, while allowing you the flexibility to control and protect your content. Additionally, the built-in resiliency and scalability of MediaPackage means that you have the right amount of resources at the right time, with no manual intervention required.

Note

This user guide is intended for creating MediaPackage resources in MediaPackage Version 2 (v2) starting from May 2023. To get started with MediaPackage v2, create your MediaPackage resources. There isn't an automated process to migrate your resources from MediaPackage v1 to MediaPackage v2.

The names of the entities that you use to access your MediaPackage resources, like URLs and ARNs, all include "mediapackagev2", to distinguish from the prior version. If you used MediaPackage prior to this release, you can't use the MediaPackage v2 AWS CLI or the MediaPackage v2 API to access any MediaPackage v1 resources.

If you created resources in MediaPackage v1, use video on demand (VOD) workflows, and aren't looking to migrate to MediaPackage v2 yet, see the [AWS Elemental MediaPackage v1 User Guide](#).

Topics

- [Are you a first-time user of MediaPackage?](#)
- [AWS Elemental MediaPackage concepts and terminology](#)
- [Supported inputs and outputs](#)
- [How MediaPackage works](#)
- [Supported features of AWS Elemental MediaPackage](#)
- [Services related to AWS Elemental MediaPackage](#)
- [Accessing MediaPackage](#)

- [Pricing for MediaPackage](#)
- [Regions for MediaPackage](#)

Are you a first-time user of MediaPackage?

If you're a first-time user of MediaPackage, we recommend that you begin by reading the following sections:

- [AWS Elemental MediaPackage concepts and terminology](#)
- [How MediaPackage works](#)
- [Supported features of AWS Elemental MediaPackage](#)
- [Getting started with AWS Elemental MediaPackage](#)

AWS Elemental MediaPackage concepts and terminology

The following are AWS Elemental MediaPackage concepts and terms to be familiar with.

Channel group

A *channel group* is the top-level resource that consists of channels and origin endpoints that are associated with it and that provides predictable URLs for stream delivery. All channels and origin endpoints within the channel group are guaranteed to share the DNS.

Channel

A *channel* represents the entry point for a content stream into MediaPackage. Upstream encoders such as AWS Elemental MediaLive send content to the channel. When MediaPackage receives a content stream, it packages the content and outputs the stream from an endpoint that you create on the channel. There's one channel for each incoming set of adaptive bitrate (ABR) streams.

Endpoint

An *endpoint* is part of a channel and represents the packaging aspect of MediaPackage. When you create an endpoint on a channel, you indicate what streaming format, packaging parameters, and features the output stream will use. Downstream devices request content from the endpoint. A channel can have multiple endpoints.

Just-in-time packaging

MediaPackage performs *just-in-time packaging* (JITP). When a playback device requests content, MediaPackage dynamically customizes the live video streams and creates a manifest in a format that's compatible with the requesting device.

Origination service

MediaPackage is considered an *origination service* because it's the point of distribution for media content delivery.

Packager

A *packager* prepares output streams for access by different types of players. The packager type specifies the streaming format that MediaPackage delivers from the endpoint (either Apple HLS, DASH-ISO, Microsoft Smooth Streaming, or Common Media Application Format [CMAF]). Additional packager settings include buffer and update durations and manifest tag handling instructions.

A packager is a part of an origin endpoint. Each endpoint must have one, and only one, packager. To use different packager types for the same content, create multiple endpoints on the channel.

Source Content

Source contents are live streams and video files that MediaPackage ingests.

- For live video, source content comes from an upstream encoder, such as AWS Elemental MediaLive. MediaPackage supports HLS source content.

Stream

A *stream* refers to the content input and output of MediaPackage.

For live workflows, an upstream encoder sends a live stream as an input to MediaPackage to the channel. When a downstream device requests playback of the content, MediaPackage dynamically packages the stream (including specifying the packager type, adding encryption, and configuring track outputs) and delivers it to the requesting device as an output of the endpoint. An endpoint can produce multiple streams.

Track

Tracks make up the output content stream. MediaPackage includes selected video, audio, and subtitles or captions tracks in the output stream. The stream delivers the tracks to the player

(either directly or through a CDN), and the player plays back the tracks based on player logic or network conditions (such as available bandwidth).

Supported inputs and outputs

This section describes the input types, input codecs, and output codecs that AWS Elemental MediaPackage supports for live content.

Topics

- [Supported input types](#)
- [Supported input codecs](#)
- [Supported output codecs](#)

The following sections describe supported input types and codecs for live streaming content.

Supported input types

Use the following input types to push streams from an external source or encoder (such as AWS Elemental MediaLive) using the HTTPS protocol:

- HLS
- CMAF

For information about CMAF ingest, see [CMAF ingest](#).

The following are additional input requirements:

- You must define a channel policy to enable content to flow into your channel from sources outside of your account.
- Media segments must not be encrypted.
- Streams can contain either muxed video and audio tracks, or unmuxed tracks.
- The input must contain at least one video track. MediaPackage doesn't support inputs that contain no video track.

Supported input codecs

These are the video, audio, and subtitles codecs that MediaPackage supports for source content streams.

Input type	Media container	Video codecs	Audio codecs	Subtitles/ captions format
HLS	- Video: TS - Audio: TS, AAC, AC3, or EC3	- H.264 (AVC) - H.265 (HEVC) with HDR-10 or Dolby Vision Profile 8.1 support	- AAC - Dolby Digital - Dolby Digital Plus	- WebVTT - CEA-608 and CEA-708 closed captions
CMAF	CMAF	- H.264 (AVC) - H.265 (HEVC) with HDR-10 or Dolby Vision Profile 8.1 support - AV1	- AAC - Dolby Digital - Dolby Digital Plus	- TTML - CEA-608 and CEA-708 closed captions

Dynamic Multiview channels

MediaPackage also supports a third channel type, **Dynamic Multiview**, which takes no input of its own. A multiview channel composites several source channels — each of which uses CMAF ingest — into a single tiled stream. For more information, see [Dynamic Multiview](#).

Supported output codecs

These are the video, audio, and subtitles codecs that MediaPackage supports when delivering live content.

Note

The AV1 video codec is supported only with CMAF endpoint types. If you configure a TS endpoint on a channel with AV1 streams those streams won't show up on the endpoint.

Endpoint type	Manifest format	Media container	Video codecs	Audio codecs	Subtitles /captions format
TS	HLS	• Video: TS • Audio: TS or AAC	• H.264 (AVC) • H.265 (HEVC) with HDR-10 or Dolby Vision Profile 8.1 support	• AAC • Dolby Digital • Dolby Digital Plus	• WebVTT
ISM	MSS (Smooth Streaming)	fragmented MP4 (PIFF/ISMV)	• H.264 (AVC) • H.265 (HEVC)	• AAC • Dolby Digital • Dolby Digital Plus	TTML
CMAF	HLS	CMAF	• H.264 (AVC) • H.265 (HEVC) with HDR-10 or Dolby Vision	• AAC • Dolby Digital • Dolby Digital Plus	• WebVTT • CEA-608 and CEA-708 closed captions

Endpoint type	Manifest format	Media container	Video codecs	Audio codecs	Subtitles /captions format
			Profile 8.1 support • AV1		
CMAF	DASH	CMAF	• H.264 (AVC) • H.265 (HEVC) with HDR-10 or Dolby Vision Profile 8.1 support • AV1	• AAC • Dolby Digital • Dolby Digital Plus	• TTML • CEA-608 and CEA-708 closed captions

How MediaPackage works

AWS Elemental MediaPackage uses just-in-time format conversion to deliver over-the-top (OTT) video from a single source to a wide variety of playback devices or content delivery networks (CDNs).

In the processing flow for live content, encoders send live HLS streams to MediaPackage. MediaPackage then packages the content, formatting it in response to playback requests from downstream devices.

The following sections describe the live processing flows.

Topics

- [General AWS Elemental MediaPackage live processing flow](#)
- [Live input redundancy AWS Elemental MediaPackage processing flow](#)

General AWS Elemental MediaPackage live processing flow

The following outlines the general flow of live content in MediaPackage:

1. An upstream encoder (such as AWS Elemental MediaLive) sends an HLS live stream using AWS Signature Version 4 to authorize request to your origin and your IAM channel policy. If you're using input redundancy, the encoder sends two identical HLS live streams to MediaPackage, one to each ingest domain on the channel. MediaPackage uses the stream from one ingest URL as the source content. If MediaPackage stops receiving content on the active ingest URL, it automatically switches to the other ingest URL for source content. Additionally, AWS scales resources up and down to handle the incoming traffic.

For more information, see [Live input redundancy AWS Elemental MediaPackage processing flow](#).

Note

To permit support for features like time-shifted viewing, MediaPackage stores all received content for a limited time. This stored content is only available for playback if it falls within the **startover window** that's defined on the endpoint. Stored content isn't available for playback if it's outside the startover window, or if you haven't defined a window on the endpoint. For more information, see [Time-shifted viewing with AWS Elemental MediaPackage](#).

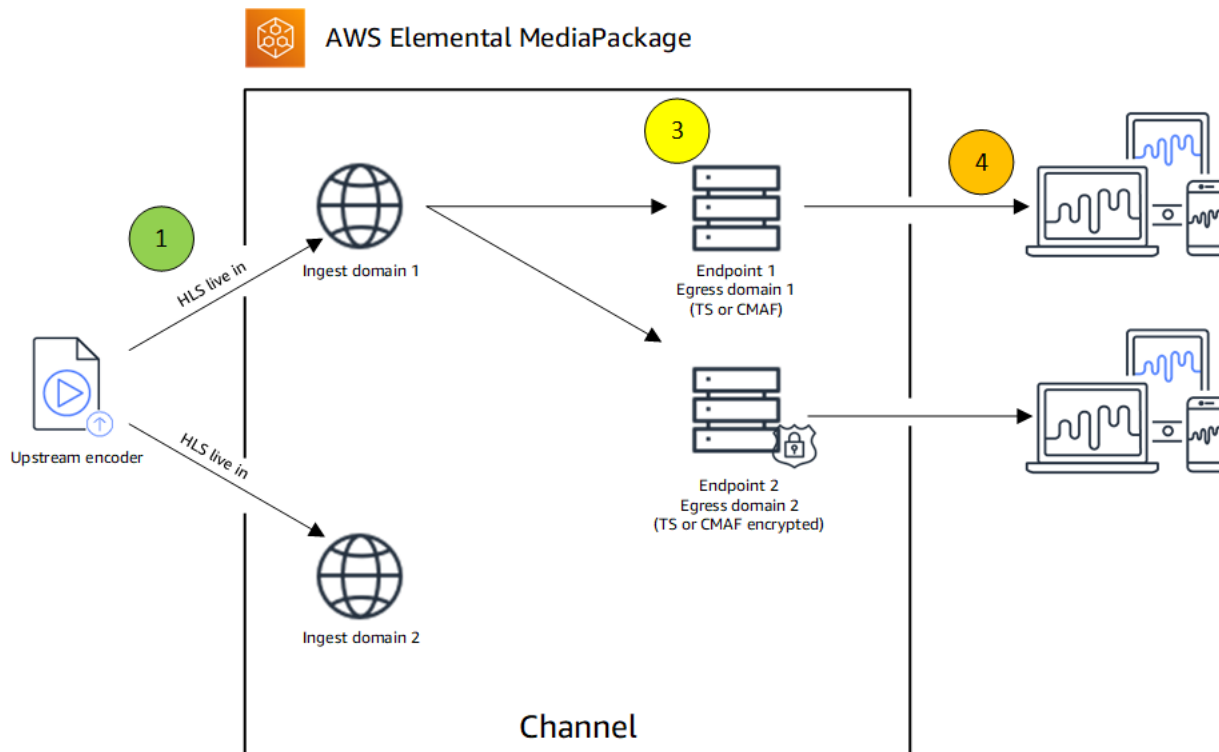
2. A downstream device requests content from MediaPackage through the endpoint egress domain. A downstream device is either a video player or a CDN. The egress domain is associated with a channel group and an endpoint for a specific streaming format (either TS or CMAF).
3. When MediaPackage receives the playback request from the downstream device, it dynamically packages the stream according to the settings that you specified on the origin endpoint. Packaging can include adding encryption and configuring audio, video, and subtitles or captions track outputs.

Be sure to order your inputs so that your preferred audio rendition is listed first in the audio section of the multivariant playlist. Do the same for the subtitles or captions. When packaging audio and subtitles or captions tracks, MediaPackage designates the first audio and captions or subtitles track as `DEFAULT=YES` and `AUTO-SELECT=YES`. This packaging overrides default and auto-select settings from the input.

4. MediaPackage delivers the output stream over HTTPS to the requesting device. As with input, AWS scales resources up and down to handle changes in traffic.

Throughout the content input and output processes, MediaPackage detects and mitigates potential infrastructure failures before they become a problem for viewers.

The following illustration shows the overall process.



Live input redundancy AWS Elemental MediaPackage processing flow

Achieve input redundancy in AWS Elemental MediaPackage by sending two streams to separate ingest domains on a channel in MediaPackage. One of the streams becomes the primary, active source of content for the endpoints, while the other continues to passively receive content. If MediaPackage stops receiving content from the active stream, it switches over to the other ingest stream so that content playback isn't interrupted.

If you use MediaPackage with AWS Elemental MediaLive (for example), here's the flow of input redundancy:

1. You create a channel group in MediaPackage, as described in [Creating a channel group in AWS Elemental MediaPackage](#). When MediaPackage provisions the channel group, it creates an egress domain for all channels and origin endpoints within the channel group.
2. You create a channel within the channel group as described in [Creating a channel in AWS Elemental MediaPackage](#). When MediaPackage provisions the channel, it creates two ingest

domains for the channel. If you're not using input redundancy, you can send a stream to either ingest domain. There's no requirement that you send content to both domains.

3. You create an origin endpoint within the channel as described in [Creating an origin endpoint in AWS Elemental MediaPackage](#).

⚠ Important

If you use short output segments, depending on your playback device, you might see buffering when MediaPackage switches inputs. You can reduce buffering by using the time delay feature on the endpoint. Be aware that using a time delay introduces latency to end-to-end delivery of the content. For information about enabling time delay, see [Creating an origin endpoint in AWS Elemental MediaPackage](#).

4. You create an input and channel in AWS Elemental MediaLive, and you add a MediaPackage output group to the channel in MediaLive. For more information, see [Creating a Channel from Scratch](#) in the *AWS Elemental MediaLive User Guide*.

If you use an HLS output group in AWS Elemental MediaLive, the input loss action on the HLS group's settings must be set to pause the output if the service doesn't receive input. If MediaLive sends a black frame or some other filler frame when it's missing input, then MediaPackage can't tell when segments are missing, and subsequently can't perform failover. For more information about setting the input loss action in MediaLive, see [Fields for the HLS Group](#) in the *AWS Elemental MediaLive User Guide*.

⚠ Important

If you use a different encoder (not AWS Elemental MediaLive) and you send two separate streams to the same channel in MediaPackage, the streams must have identical encoder settings and manifest names. Otherwise, input redundancy might not work correctly and playback could be interrupted if the inputs switch.

5. You start the channel in AWS Elemental MediaLive to send the streams to MediaPackage.
6. MediaPackage receives content on both of the ingest URLs, but only one of the streams is used for source content at a time. If the active stream is missing any segments, then MediaPackage automatically fails over to the other stream. MediaPackage continues to use this stream until failover is needed again.

The formula that's used to determine if an input is missing segments is based on the segment lengths on the inputs and the endpoints. If an input is missing segments and quickly recovers, an endpoint with longer segment lengths won't switch inputs. This might result in different endpoints on the channel using different inputs (if one endpoint switches and the other doesn't). This is expected behavior and should not affect the content workflow.

Supported features of AWS Elemental MediaPackage

MediaPackage supports the following features.

Audio

MediaPackage supports multi-language audio inputs and the following audio codecs:

- AAC stereo
- Dolby AC3 and E-AC3 (Dolby Digital and Dolby Digital+)

MediaPackage accepts these codecs from the input source and passes them through to the output stream.

Be sure to order your inputs so that your preferred audio rendition is listed first in the audio section of the multivariant playlist. When packaging audio and subtitles or captions tracks, MediaPackage designates the first audio track as `DEFAULT=YES` and `AUTO-SELECT=YES`. This packaging overrides default and auto-select settings from the input.

Important

MediaPackage doesn't support audio-only inputs. The stream configuration from the encoder must include at least one video track.

Captions

Your embedded source captions can be CEA-608 captions, CEA-708 captions, or both CEA-608 and CEA-708. MediaPackage will pass through these captions in the media segments on TS and CMAF origin endpoints, and generate the appropriate manifest signaling.

Be sure to order your inputs so that your preferred captions rendition is listed first in the captions section of the multivariant playlist. When packaging captions tracks, MediaPackage

designates the first captions track as DEFAULT=YES and AUTO-SELECT=YES. This packaging overrides default and auto-select settings from the input.

⚠ Important

Your input HLS playlist must include captions signaling tags. If not present, MediaPackage will not be able to generate the corresponding output manifest signaling.

DRM

MediaPackage supports content protection through digital rights management (DRM), including advanced features like session key exclusion for HLS manifests to improve compatibility with legacy clients and provide granular access control. For information, see [Content encryption and DRM in AWS Elemental MediaPackage](#).

HLS Rendition Groups

MediaPackage supports rendition groups for incoming and outgoing HLS content. For information about output rendition groups, see [AWS Elemental MediaPackage rendition groups reference](#).

Input Redundancy

Input redundancy is available with only live workflows in MediaPackage.

MediaPackage creates two ingest URLs on every channel group so that you can create input redundancy by sending two identical streams to the same channel. For information about how input redundancy works, see [Live input redundancy AWS Elemental MediaPackage processing flow](#).

Low-latency streaming

MediaPackage supports Apple low-latency HLS, which is a technology aimed at reducing the delay between the time content is captured and the time it is displayed on the viewer's screen. The goal is to achieve minimal end-to-end delay (or "glass-to-glass" delay) by using techniques such as parallel delivery and reduced buffering. This technology enables a more seamless and immersive real-time viewing experience for users, particularly in applications such as live video streaming, teleconferencing, and online gaming.

Subtitles

MediaPackage supports input WebVTT text-based subtitles and passes through the subtitles.

Be sure to order your inputs so that your preferred subtitles rendition is listed first in the subtitles section of the multivariant playlist. When packaging subtitles tracks, MediaPackage designates the first subtitles track as `DEFAULT=YES` and `AUTO-SELECT=YES`. This packaging overrides default and auto-select settings from the input.

Time-shift Viewing

Time-shift viewing is available with only live workflows in MediaPackage.

MediaPackage supports playback of a stream at a time earlier than the current time. Start-over, catch-up TV, and time delay are all supported. For more information about setting up time-shift capabilities, see [Time-shifted viewing with AWS Elemental MediaPackage](#).

Video

MediaPackage supports the input H.264 video codec and passes it through to the output stream. CMAF endpoints in MediaPackage also support H.265/HEVC and HDR-10, following the Apple specification to applicable playback devices.

Important

MediaPackage requires at least one video track to be present in the stream configuration from the encoder. The service doesn't support audio-only ingest.

Services related to AWS Elemental MediaPackage

You might use the following services when using MediaPackage.

- **Amazon CloudWatch** is a monitoring service for AWS Cloud resources and the applications that you run on AWS. Use CloudWatch to track metrics such as content input and output request counts. For more information, see [Amazon CloudWatch](#).
- **AWS Elemental MediaLive (MediaLive)** is a live video processing service that encodes high-quality live video streams for broadcast television and multi-screen devices. Use MediaLive to encode content streams and send them to MediaPackage for packaging. For more information about how encoders (such as MediaLive) work with MediaPackage, see [How MediaPackage works](#).
- **AWS Elemental MediaTailor (MediaTailor)** is a scalable ad insertion service that runs in the AWS Cloud. Use MediaTailor to serve targeted ads to viewers. For more information, see [AWS Elemental MediaTailor](#).

- **AWS Identity and Access Management (IAM)** is a web service that helps you securely control access to AWS resources for your users. Use IAM to control who can use your AWS resources (authentication) and what resources users can use in which ways (authorization). For more information, see [Setting up MediaPackage](#).

Accessing MediaPackage

You can access MediaPackage using any of the following methods.

- **AWS Management Console** - The procedures throughout this guide explain how to use the AWS Management Console to perform tasks for MediaPackage.

```
https://console.aws.amazon.com/mediapackage/
```

- **AWS Command Line Interface** - For more information, see the [AWS Command Line Interface User Guide](#).

```
aws mediapackagev2
```

- **MediaPackage API** - For information about API actions and about how to make API requests, see the [AWS Elemental MediaConnect API Reference](#).

```
https://mediapackagev2.region.amazonaws.com
```

- **AWS SDKs** - If you're using a programming language that AWS provides an SDK for, you can use an SDK to access MediaPackage. SDKs simplify authentication, integrate easily with your development environment, and provide easy access to MediaPackage commands. For more information, see [Tools for Amazon Web Services](#).
- **AWS Tools for Windows PowerShell** - For more information, see the [AWS Tools for PowerShell User Guide](#).

Pricing for MediaPackage

As with other AWS products, there are no contracts or minimum commitments for using MediaPackage. You're charged only for AWS resources that your account uses. Pricing is pay-as-you-go and consists of the following:

- A per GB charge for received content

- A per GB charge for content that's streamed out of MediaPackage

Content that's cached and served from a content delivery network (CDN) doesn't incur this per GB charge.

For detailed pricing information, see [MediaPackage Pricing](#).

Regions for MediaPackage

To reduce latency in your applications, MediaPackage offers a regional endpoint for your requests. To view the list of AWS Regions where MediaPackage is available, see [MediaPackage Regions](#).

MediaPackage V2 control plane APIs support IPv6 in all supported regions. For more information, see [IPv6 support](#).

AWS opt-in Regions

Although most AWS Regions are active by default for your AWS account, certain Regions are activated only when you manually select them. This document refers to those Regions as *opt-in Regions*. In contrast, Regions that are active by default, as soon as your AWS account is created, are referred to as *commercial Regions*, or simply, *Regions*.

The term *opt-in* has a historical basis. Any AWS Regions introduced after March 20, 2019 are considered to be opt-in Regions. Opt-in Regions have higher security requirements than commercial Regions, regarding the sharing of IAM data through accounts that are active in opt-in Regions. All of the data managed through the IAM service is considered identity data, including users, groups, roles, policies, identity providers, their associated data (for example, X.509 signing certificates or context-specific credentials), and other account-level settings, such as password policy and the account alias.

You can activate opt-in Regions automatically during channel setup, by selecting them. Your channel becomes active in all selected Regions.

If you choose to select an opt-in Region as for your MediaPackage resources, enable it first by following the steps in [Enabling a Region](#), when signed in to the AWS Management Console.

MediaPackage is available in the following AWS opt-in Regions:

- Middle East (UAE) Region, me-central-1

- Asia Pacific (Hyderabad) Region, ap-south-2
- Asia Pacific (Melbourne) Region, ap-southeast-4

Setting up MediaPackage

This section provides procedures to set up your organization to use AWS Elemental MediaPackage. It also provides information about determining the IAM permissions that users and other AWS identities require. These permissions let you impose restricted controls on users and other AWS identities, in conformance with the security policies and procedures of your organization.

Topics

- [Sign up for an AWS account](#)
- [Set up additional IAM permissions](#)
- [Allowing MediaPackage to access other AWS services](#)
- [Download tools](#)

Sign up for an AWS account

To get started with AWS, you need an AWS account. For information about creating an AWS account, see [Getting started with an AWS account](#) in the *AWS Account Management Reference Guide*.

Set up additional IAM permissions

By default, users and roles don't have permission to create or modify MediaPackage resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see [Create IAM policies \(console\)](#) in the *IAM User Guide*.

For details about actions and resource types defined by MediaPackage, including the format of the ARNs for each of the resource types, see [Actions, resources, and condition keys for AWS Elemental MediaPackage](#) in the *Service Authorization Reference*.

This section describes the permissions that you must assign to users and other AWS identities so that they can work with MediaPackage and other AWS services that your workflows use. After you have identified the required permissions, you will be able to design and create the relevant policies, and attach those policies to groups of users or to roles.

This section assumes that you have already performed these tasks:

- You have signed up for MediaPackage and created an administrator.
- You have read the recommendations in [Identity and Access Management for AWS Elemental MediaPackage](#) about how to create administrators, users, and other AWS identities.

Topics

- [Create a role in the IAM console](#)
- [Assume the role from the IAM console or AWS CLI](#)
- [Add permissions for tagging](#)

Create a role in the IAM console

Create a role in the IAM console for each policy that you create. This allows users to assume a role rather than attaching individual policies to each user.

To create a role in the IAM console

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
3. Under **Select trusted entity**, choose **AWS account**.
4. Under **An AWS account**, select the account with the users that will be assuming this role.
 - If a third-party will be accessing this role, it's best practice to select **Require external ID**. For more information about external IDs, see [Using an external ID for third-party access](#) in the *IAM User Guide*.
 - It's best practice to require multi-factor authentication (MFA). You can select the check box next to **Require MFA**. For more information about MFA, see [Multi-factor authentication \(MFA\)](#) in the *IAM User Guide*.
5. Choose **Next**.
6. Under **Permissions policies**, search for and add the policy with the appropriate MediaPackage permissions level.
 - For access to live functionality, choose one of the following options:

- Use **AWSElementalMediaPackageFullAccess** to allow the user to perform all actions on all live resources in MediaPackage.
 - Use **AWSElementalMediaPackageReadOnly** to provide the user read-only rights for all live resources in MediaPackage.
7. Add policies to allow the MediaPackage console to make calls to Amazon CloudWatch on the user's behalf. Without these policies, the user is able to use the service's API only (not the console). Choose one of the following options:
 - Use **ReadOnlyAccess** to allow MediaPackage to communicate with CloudWatch, and also provide the user read-only access to all AWS services on your account.
 - Use **CloudWatchReadOnlyAccess**, **CloudWatchEventsReadOnlyAccess**, and **CloudWatchLogsReadOnlyAccess** to allow MediaPackage to communicate with CloudWatch, and limit the user's read-only access to CloudWatch.
 8. (Optional) Set a [permissions boundary](#). This is an advanced feature that is available for service roles, but not service-linked roles.
 1. Expand the **Permissions boundary** section and choose **Use a permissions boundary to control the maximum role permissions**. IAM includes a list of the AWS managed and customer managed policies in your account.
 2. Select the policy to use for the permissions boundary or choose **Create policy** to open a new browser tab and create a new policy from scratch. For more information, see [Creating IAM policies](#) in the *IAM User Guide*.
 3. After you create the policy, close that tab and return to your original tab to select the policy to use for the permissions boundary.
 9. Verify that the correct policies are added to this group, and then choose **Next**.
 10. If possible, enter a role name or role name suffix to help you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both **PRODRole** and **prodrole**. Because various entities might reference the role, you cannot edit the name of the role after it has been created.
 11. (Optional) For **Description**, enter a description for the new role.
 12. Choose **Edit** in the **Step 1: Select trusted entities** or **Step 2: Select permissions** sections to edit the use cases and permissions for the role.
 13. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see [Tagging IAM resources](#) in the *IAM User Guide*.

14. Review the role and then choose **Create role**.

Assume the role from the IAM console or AWS CLI

View the following resources for learning about granting permissions for users to assume the role and how users can switch to the role from the IAM console or AWS CLI.

- For more information about granting a user permissions to switch roles, see [Granting a user permissions to switch roles](#) in the *IAM User Guide*.
- For more information about switching roles (console), see [Switching to a role \(console\)](#) in the *IAM User Guide*.
- For more information about switching roles (AWS CLI), see [Switching to an IAM role \(AWS CLI\)](#) in the *IAM User Guide*.

Add permissions for tagging

When users create channel groups, channels, or origin endpoints, they can optionally attach tags to the resource during creation. Typically, your organization has a policy to tag or to omit tags. There are two services that control permissions for tagging, for two different scenarios:

- The ability to tag during channel creation is controlled by actions within MediaPackage.
- The ability to modify tags in existing resources is controlled by actions within Resource Group Tagging. See [Working with Tag Editor](#) in [Getting Started with the AWS Management Console](#).

Allowing MediaPackage to access other AWS services

Some features require you to allow MediaPackage to access other AWS services, such as Amazon S3 and AWS Secrets Manager (Secrets Manager). To allow this access, create an IAM role and policy with the appropriate permissions. The following steps describe how to create roles and policies for MediaPackage features.

Steps

- [Step 1: Create a policy](#)
- [Step 2: Create a role](#)
- [Step 3: Modify the trust relationship](#)

Step 1: Create a policy

The IAM policy defines the permissions that AWS Elemental MediaPackage (MediaPackage) requires to access other services.

- For live-to-VOD workflows, create a policy that allows MediaPackage to read from the Amazon S3 bucket and store the live-to-VOD asset in it.
- For content delivery network (CDN) authorization with static headers, create a policy that allows MediaPackage to read from a secret in Secrets Manager and a key in AWS Key Management Service (AWS KMS). This policy is *not* needed if you're using AWS Signature Version 4 (SigV4) authentication.

Use the following instructions to set up the policies that you need.

Amazon S3 access for live-to-VOD workflows

If you use MediaPackage to harvest a live-to-VOD asset from a live stream, you need a policy that allows you to do these things in Amazon S3:

- `PutObject`: MediaPackage can save the VOD asset in the bucket.
- `GetBucketLocation`: MediaPackage can retrieve the Region for the bucket. The bucket must be in the same AWS Region as the MediaPackage VOD resources.

To use the JSON policy editor to create a policy

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane on the left, choose **Policies**.

If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.

3. At the top of the page, choose **Create policy**.
4. In the **Policy editor** section, choose the **JSON** option.
5. Enter the following JSON policy document:

```
{  
  "Version": "2012-10-17",
```

```
"Statement": [
  {
    "Action": [
      "s3:PutObject",
      "s3:ListBucket",
      "s3:GetBucketLocation"
    ],
    "Resource": [
      "arn:aws:s3:::bucket_name/*",
      "arn:aws:s3:::bucket_name"
    ],
    "Effect": "Allow"
  }
]
```

6. Choose **Next**.

 Note

You can switch between the **Visual** and **JSON** editor options anytime. However, if you make changes or choose **Next** in the **Visual** editor, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy restructuring](#) in the *IAM User Guide*.

7. On the **Review and create** page, enter a **Policy name** and a **Description** (optional) for the policy that you are creating. Review **Permissions defined in this policy** to see the permissions that are granted by your policy.
8. Choose **Create policy** to save your new policy.

Secrets Manager and AWS KMS access for CDN authorization

If you use content delivery network (CDN) authorization headers to restrict access to your endpoints in MediaPackage, you need a policy that allows you to do these things in Secrets Manager:

- **GetSecretValue** - MediaPackage can retrieve the encrypted authorization code from a version of the secret that's in Secrets Manager.
- **DescribeSecret** - MediaPackage can retrieve the details of the secret from Secrets Manager, excluding encrypted fields.

- **BatchGetSecretValue** - MediaPackage can retrieve a list of secrets from Secrets Manager.

The following permissions are required only if you customer-managed AWS KMS key. If you use the default key that AWS KMS creates, you don't need to manually add permissions. AWS KMS automatically adds the appropriate permissions for default keys.

- **Decrypt**: MediaPackage can decrypt the key from AWS KMS.
- **DescribeKey**: MediaPackage can retrieve the details of the key from AWS KMS.

To use the JSON policy editor to create a policy

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation column on the left, choose **Policies**.

If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.

3. At the top of the page, choose **Create policy**.
4. Choose the **JSON** tab.
5. Enter the following JSON policy document, replacing *region*, *account-id*, *secret-name*, and *key-name* with your own information:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:secret-name-AbCDeF"
    },
    {
      "Effect": "Allow",
```

```
        "Action": [
            "secretsmanager:BatchGetSecretValue"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "kms:Decrypt",
            "kms:DescribeKey"
        ],
        "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-name"
    }
]
```

6. Choose **Review policy**.

 Note

You can switch between the **Visual editor** and **JSON** tabs any time. However, if you make changes or choose **Review policy** in the **Visual editor** tab, IAM might restructure your policy to optimize it for the visual editor. For more information, see [Policy restructuring](#) in the *IAM User Guide*.

7. On the **Review policy** page, enter a **Name** and an optional **Description** for the policy that you are creating. Review the policy **Summary** to see the permissions that are granted by your policy. Then choose **Create policy** to save your work.

Step 2: Create a role

An [IAM role](#) is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user in that it is an AWS identity with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session.

Create a role that AWS Elemental MediaPackage assumes when ingesting source content or reading secrets and keys for CDN authorization. When you create the role, MediaPackage isn't available to pick as the trusted entity to assume the role. Choose Amazon Elastic Compute Cloud (Amazon EC2) temporarily instead. In the [next step](#), you change the trusted entity to MediaPackage.

For information about creating a service role, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.

Step 3: Modify the trust relationship

The trust relationship defines what entities can assume the role that you created in [the section called "Step 2: Create a role"](#). When you created the role and established the trusted relationship, you chose Amazon EC2 as the trusted entity. Modify the role so that the trusted relationship is between your AWS account and AWS Elemental MediaPackage.

To change the trust relationship to MediaPackage

1. Access the role that you created in [the previous step](#).

If you're not already displaying the role, in the navigation pane of the IAM console, choose **Roles**. Search for and choose the role that you created.

2. On the **Summary** page for the role, choose **Trust relationships**.
3. Choose **Edit trust relationship**.
4. On the **Edit Trust Relationship** page, in the **Policy Document**, change `ec2.amazonaws.com` to `mediapackagev2.amazonaws.com`.

The policy document should now look like this:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "mediapackagev2.amazonaws.com"
      },
    },
  ],
}
```

```

        "Action": "sts:AssumeRole"
      }
    ]
  }

```

If you're using MediaPackage and related services in an opt-in Region, the Region must be listed in the **Service** section of the policy document. For example, if you're using services in the Asia Pacific (Melbourne) Region, the policy document looks like this:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": ["mediapackagev2.amazonaws.com", "mediapackagev2.ap-
southeast-4.amazonaws.com"]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

For a list of opt-in Regions, see [AWS opt-in Regions](#).

5. Choose **Update Trust Policy**.
6. On the **Summary** page, make a note of the value in **Role ARN**. You use this ARN when you ingest source content for video on demand (VOD) workflows or set up CDN authorization. The ARN looks like this:

arn:aws:iam::*111122223333*:role/*role-name*

In the example, *111122223333* is your AWS account number.

Download tools

The AWS Management Console includes a console for MediaPackage, but if you want to access the services programmatically, see the following:

- The API guides document the operations that the services support and provide links to the related SDK and CLI documentation:
 - [AWS Elemental MediaPackage API Reference](#)
- To call an API without having to handle low-level details like assembling raw HTTP requests, you can use an AWS SDK. The AWS SDKs provide functions and data types that encapsulate the functionality of AWS services. To download an AWS SDK and access installation instructions, see the applicable page:
 - [Go](#)
 - [JavaScript](#)
 - [.NET](#)
 - [Node.js](#)
 - [Python](#)
 - [Ruby](#)

For a complete list of AWS SDKs, see [Tools for Amazon Web Services](#).

- You can use the AWS Command Line Interface (AWS CLI) to control multiple AWS services from the command line. You can also automate your commands using scripts. For more information, see [AWS Command Line Interface](#).
- AWS Tools for Windows PowerShell supports these AWS services. For more information, see [AWS Tools for PowerShell Cmdlet Reference](#).

IPv6 support for AWS Elemental MediaPackage V2 control plane

AWS Elemental MediaPackage V2 control plane APIs support dual-stack (IPv4 and IPv6) endpoints. This enables you to make API requests using either IPv4 or IPv6 protocols for management operations such as creating channel groups, channels, and origin endpoints.

Note

IPv6 support applies to control plane operations only. Content ingest and delivery endpoints continue to use IPv4. IPv6 support for the data plane is planned for a future release.

IPv6 endpoints

MediaPackage V2 provides the following dual-stack endpoints:

- **Standard endpoint:** `mediapackagev2.region.api.aws`
- **FIPS endpoint** (FIPS-enabled regions): `mediapackagev2-fips.region.api.aws`

Your existing applications continue to work with the original IPv4-only endpoints (`mediapackagev2.region.amazonaws.com`). For a complete list of available endpoints, see [MediaPackage endpoints and quotas](#) in the *AWS General Reference*.

Using IPv6 endpoints

To use the IPv6 endpoints, specify the dual-stack endpoint URL when making API calls.

AWS CLI example:

```
aws mediapackagev2 list-channel-groups \  
  --endpoint-url https://mediapackagev2.us-east-1.api.aws
```

SDK example (Python):

```
import boto3

client = boto3.client(
    'mediapackagev2',
    endpoint_url='https://mediapackagev2.us-east-1.api.aws'
)
```

Getting started with AWS Elemental MediaPackage

This tutorial describes how to get started with MediaPackage, using the console to create a channel and endpoints for streaming live videos.

Topics

- [Prerequisites](#)
- [Step 1: Access MediaPackage](#)
- [Step 2: Create a channel group](#)
- [Step 3: Create a channel](#)
- [Step 4: Create an endpoint](#)
- [Step 5: Clean up](#)

Prerequisites

Before you can use MediaPackage, you need an AWS account and the appropriate permissions to access, view, and edit MediaPackage components. Make sure that your system administrator has completed the steps in [Setting up MediaPackage](#), and then return to this tutorial.

For supported live inputs and codecs, see [Supported inputs and outputs](#).

Step 1: Access MediaPackage

Using your IAM credentials, sign in to the AWS Elemental MediaPackage console:

```
https://console.aws.amazon.com/mediapackage/
```

Step 2: Create a channel group

A channel group is the top-level resource that streamlines the organization of multiple channels and origin endpoints associated with it.

To create a channel group

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.

2. On the **Channel groups** page, choose **Create channel group**.
3. Enter a unique name that describes the channel group and, optionally, a description.
4. Choose **Create**.

MediaPackage displays the new channel group's details page.

Step 3: Create a channel

The channel represents the input to MediaPackage for incoming live content from an encoder such as AWS Elemental MediaLive. The channel receives content, and after packaging it, outputs it through an endpoint to downstream devices (such as video players or CDNs) that request the content.

MediaPackage does not require that you supply any customer data. There are no fields in channels where there is an expectation that you will provide customer data.

To create a channel

1. Access the channel group that the channel will be associated with.
2. In the **Channel group details** page, under **Channels**, choose **Create channel**.
3. Enter a unique name that describes the channel and, optionally, a description.
4. Choose your channel's IAM policy that defines the permissions of your channel.
5. Choose **Create**.

MediaPackage displays the new channel's details page. The channel is active and can start receiving content as soon as it's created.

Step 4: Create an endpoint

The endpoint is attached to a channel, and represents the output of the live content. You can associate multiple endpoints to a single channel. Each endpoint gives players and downstream CDNs (such as Amazon CloudFront) access to the content for playback.

To create an endpoint

1. On the **Channels** page, choose the channel that the endpoint will be associated with.
2. On the details page for the channel, under **Origin endpoints**, choose **Create endpoint**.

3. Enter a unique name that describes the endpoint and, optionally, a description.
4. Choose the container type and define the corresponding settings.
5. Choose your origin endpoint's IAM policy that defines the permissions of your endpoint.
6. Define the manifests emitted from the origin endpoint.
7. Choose **Save**.

MediaPackage displays the channel's details page, including the endpoint that you just created.

Step 5: Clean up

To avoid extraneous charges, be sure to delete all unnecessary channel groups, channels, and endpoints. You must delete the channels and endpoints before you can delete the channel group.

1. Delete the endpoint as described in [Deleting an endpoint in AWS Elemental MediaPackage](#).
2. Delete a channel as described in [Deleting a channel in AWS Elemental MediaPackage](#).
3. Delete the channel group as described in [Deleting a channel group from AWS Elemental MediaPackage](#).

Access control best practices for AWS Elemental MediaPackage

MediaPackage provides a variety of security features and tools. The following topics describe some tools and settings that you might want to use to help control access when performing certain tasks or operating in specific environments. Proper application of these tools can help maintain the integrity of your data and help ensure that your resources are accessible to the intended users.

Topics

- [Creating new resources](#)
- [Sharing resources](#)
- [Protecting data](#)

Creating new resources

When creating new resources, you should apply the following tools and settings to help ensure that your MediaPackage resources are protected.

Grant access with IAM identities

When setting up accounts for new team members who require MediaPackage access, use IAM users and roles to ensure least privileges. You can also implement a form of IAM multi-factor authentication (MFA) to support a strong identity foundation. Using IAM identities, you can grant unique permissions to users and specify what resources they can access and what actions they can take. IAM identities provide increased capabilities, including the ability to require users to enter login credentials before accessing shared resources and apply permission hierarchies to different objects within a single bucket.

For more information, see [Identity and Access Management for AWS Elemental MediaPackage](#).

Resource policies

With resource policies, you can personalize channel and origin endpoint access to help ensure that only those users you have approved can access resources and perform actions within them. In addition to resource policies, you should use resource-level Block Public Access settings to further limit public access to your data.

For more information, see [Resource-based policy examples](#).

When creating policies, avoid the use of wildcard characters in the `Principal` element because it effectively allows anyone to access your MediaPackage resources. It's better to explicitly list users or groups that are allowed to access the resource. Rather than including a wildcard for their actions, grant them specific permissions when applicable.

To further maintain the practice of least privileges, `Deny` statements in the `Effect` element should be as broad as possible and `Allow` statements should be as narrow as possible. `Deny` effects paired with the `"mediapackagev2: *"` action are another good way to implement opt-in best practices for the users included in policy condition statements.

Sharing resources

There are several different ways that you can share resources with a specific group of users. You can use the following tools to share a set of documents or other resources to a single group of users, department, or an office. Although they can all be used to accomplish the same goal, some tools might pair better than others with your existing settings.

User policies

You can share resources with a limited group of people using IAM groups and user policies. When creating a new IAM user, you are prompted to create and add them to a group. However, you can create and add users to groups at any point. If the individuals you intend to share these resources with are already set up within IAM, you can add them to a common group and share the bucket with their group within the user policy. You can also use IAM user policies to share individual objects within a bucket.

For more information, see [Identity-based policy examples for MediaPackage](#).

Tagging

If you use object tagging to categorize storage, you can share objects that have been tagged with a specific value with specified users. Resource tagging allows you to control access to objects based on the tags associated with the resource that a user is trying to access. To do this, use the `ResourceTag/key-name` condition within an IAM user policy to allow access to the tagged resources.

For more information, see [Controlling access to AWS resources using resource tags](#) in the *IAM User Guide*.

Protecting data

Use the following tools to help protect data in transit and at rest, both of which are crucial in maintaining the integrity and accessibility of your data.

Signing methods

AWS Signature Version 4 is the process of adding authentication information to AWS requests sent by HTTP. For security, most requests to AWS must be signed with an access key, which consists of an access key ID and secret access key. These two keys are commonly referred to as your security credentials. For more information, see [Authenticating Requests \(AWS Signature Version 4\)](#) and [Signing AWS API requests](#) in the *IAM User Guide*.

Delivering live content from AWS Elemental MediaPackage

AWS Elemental MediaPackage uses the following resources for live content:

- A *channel group* is the top-level resource that consists of channels and origin endpoints that are associated with it and that provides predictable URLs for stream delivery. All channels and origin endpoints within the channel group are guaranteed to share the DNS.
- A *channel* is the entry point for your live streams from upstream encoders.

For supported live inputs and codecs, see [Supported inputs and outputs](#).

- An *origin endpoint* tells MediaPackage how to package outbound content. Endpoints are associated with channels and hold encryption, stream, and packaging settings.

The following sections describe how to use these resources to manage live content in MediaPackage.

Topics

- [Working with channel groups in AWS Elemental MediaPackage](#)
- [Working with channels in AWS Elemental MediaPackage](#)
- [Working with origin endpoints in AWS Elemental MediaPackage](#)

Working with channel groups in AWS Elemental MediaPackage

A channel group is the top-level resource that consists of channels and origin endpoints associated with it. After you create a channel group, MediaPackage provides a fixed egress domain for its lifetime, regardless of any failures or upgrades that might occur. All channels and origin endpoints belonging to this channel group use the same egress domain. Direct your CDNs to this domain for stream delivery from MediaPackage.

For each channel group, you add channels that define the entry point for a content stream into MediaPackage. You then add origin endpoints to the channels that define the packaging options for the output stream.

Topics

- [Creating a channel group in AWS Elemental MediaPackage](#)
- [Viewing channel group details in AWS Elemental MediaPackage](#)
- [Editing a channel group in AWS Elemental MediaPackage](#)
- [Deleting a channel group from AWS Elemental MediaPackage](#)

Creating a channel group in AWS Elemental MediaPackage

This guide shows how to create a channel group as a holder for your channels and origin endpoints. You can provide high-level information about your channel group and can add a certain number of channel groups for each account. After you create a channel group, you can add channels to the channel group.

You can use the MediaPackage console, MediaPackage API, or AWS Command Line Interface (AWS CLI) to create a channel group. When you're creating a channel group, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn't require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

To create a channel group

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.
2. Choose **Create channel group** from the **Channel groups** list.
3. For **Name**, enter a name that describes the channel group. This is the name that you use for API and console interactions. The name is the primary identifier for the channel group, and must be unique for your account in the AWS Region. Supported characters are **A-Z**, **a-z**, **0-9**, **_** (underscore), and **-** (hyphen) with a length of 1–256 characters. You can't use spaces in the name, and you can't change the name after you create the channel group.
4. (Optional) For **Description**, enter any descriptive text that helps you to identify the channel group.
5. Choose **Create**.

MediaPackage displays the new channel group's details page.

After you create a channel group, MediaPackage provides an egress domain URL that is fixed for the lifetime of the channel group. This domain remains regardless of any failures or upgrades that might happen over time.

All channels and origin endpoints that belong to this channel group will use the same domain URL. For stream delivery from MediaPackage, direct your CDNs to this domain.

When you create a channel group, if you exceed the quotas on the account, you'll receive an error. The error will be similar to Too many requests, please try again. Resource limit exceeded. This error means that either you exceeded the API request quotas, or that you reached the maximum number of channel groups that your account permits.

You can add channels to a channel group to do the following:

- Permit a live content stream from a source such as AWS Elemental MediaLive or another encoder.
- Permit downstream video players and content delivery networks (CDNs) to start requesting content playback.

For instructions on adding channels to a channel group from the MediaPackage console, see [the section called "Working with channels"](#).

Viewing channel group details in AWS Elemental MediaPackage

This guide shows how to view all channel groups that are configured in AWS Elemental MediaPackage. You can also view the details of a specific channel group. This includes the channels and origin endpoints that are associated with it. You can use the MediaPackage console, MediaPackage API, or AWS CLI to view channel group details.

To view a channel group

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.

The console shows all existing channel groups that are configured in MediaPackage.

2. (Optional) To adjust your viewing preferences, choose **Preferences**. For example, you can adjust the page size and properties that you want to view.
3. To view more information about a specific channel group, select that channel group from the **Channel groups** list.

MediaPackage displays important information such as the values for egress domain, when the channel group was created, the ARN, and any channels that are associated with the channel group.

Editing a channel group in AWS Elemental MediaPackage

This guide shows how to edit the description on a channel group for easier identification later from the AWS Elemental MediaPackage console. You can't edit the name of the channel group.

You can use the MediaPackage console, MediaPackage API, or AWS CLI to edit a channel group. When you're editing a channel group, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn't require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

To edit a channel group

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.

The console shows all existing channel groups that are configured in MediaPackage.

2. Select the name of the channel group that you want to edit.
3. On the channel group's details page, choose **Edit**.
4. Edit the description for easier identification later.
5. Choose **Edit**.

Deleting a channel group from AWS Elemental MediaPackage

This guide shows how to delete a channel group to stop AWS Elemental MediaPackage from receiving content. Before you can delete the channel group, you must delete the channel group's channels and endpoints. For instructions, see [Deleting a channel in AWS Elemental MediaPackage](#) and [Deleting an endpoint in AWS Elemental MediaPackage](#). You can use the MediaPackage console, MediaPackage API, or AWS CLI to delete a channel group.

⚠ Warning

If you delete a channel group, you'll lose access to the egress domain URL. If that happens, you must create a new channel group to replace it.

To delete a channel group

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.

The console shows all existing channel groups that are configured in MediaPackage.

2. Select the name of the channel group that you want to delete.
3. Choose **Delete**.
4. Choose **Delete** in the confirmation dialog box.

Working with channels in AWS Elemental MediaPackage

A channel is part of a channel group and represents the entry point for a content stream into MediaPackage. After you create a channel, MediaPackage provides ingest endpoint domains for its lifetime, regardless of any failures or upgrades that might occur.

Upstream encoders such as AWS Elemental MediaLive send content to the channel. When MediaPackage receives a content stream, it packages the content and outputs the stream from an origin endpoint that you create on the channel. Each incoming set of adaptive bitrate (ABR) streams has one channel. A channel group can have multiple channels.

For supported live inputs and codecs, see [Supported inputs and outputs](#).

Topics

- [Creating a channel in AWS Elemental MediaPackage](#)
- [Viewing channel details in AWS Elemental MediaPackage](#)
- [Editing a channel in AWS Elemental MediaPackage](#)
- [Resetting channel history in AWS Elemental MediaPackage](#)
- [Deleting a channel in AWS Elemental MediaPackage](#)

Creating a channel in AWS Elemental MediaPackage

These steps show how to create a channel in MediaPackage to start receiving content streams. Later, you add an origin endpoint to the channel. This endpoint is the access point for content playback requests. We recommend that you spread out channels between channel groups, such as putting redundant channels in the same AWS Region in different channel groups.

You can use the MediaPackage console, MediaPackage API, or AWS CLI to create a channel. When you're creating a channel, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn't require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

To create a channel

1. Access the channel group that the channel will be associated with, as described in [Viewing channel group details in AWS Elemental MediaPackage](#).
2. Choose **Create channel** from the **Channels** list.
3. For **Name**, enter a name that describes the channel. The name is the primary identifier for the channel, and must be unique for your account in the AWS Region and channel group. Supported characters are **A-Z, a-z, 0-9, _** (underscore), and **-** (hyphen) with a length of 1 to 256 characters. You can't use spaces in the name, and you can't change the name after you create the channel.

The name is the primary identifier for the channel, and it must be unique for your account in the AWS Region and channel group.

4. (Optional) For **Description**, enter descriptive text to help you identify the channel.
5. Choose the **Input type** that you want to use for your channel:
 - **HLS** input type requires your live encoder to produce HLS with TS streams and to ingest it onto MediaPackage using plain HTTP PUT requests.
 - **CMAF** requires your live encoder to follow the [DASH-IF Live Media Ingest Protocol version 1.2](#), producing and ingesting CMAF streams using Interface-1 (CMAF Ingest, described in section 6).

MediaPackage CMAF Ingest has currently been tested with MediaLive and the AWS Elemental Live encoder. A detailed specification of the MediaPackage encoder requirements

for CMAF ingest is available upon request to live-encoder manufacturers who would want to validate interoperability of their solution with MediaPackage CMAF Ingest.

- **Dynamic Multiview** creates a channel that composites several source channels into a single tiled stream, server-side. A multiview channel receives no ingest of its own; the source channels it combines must use CMAF ingest. For more information, see [Dynamic Multiview](#).

Important

Once you will have created your channel, you will not be able to change its input type. Certain features, like cross-region failover support, are available only on channels using CMAF ingest - and more upcoming features will also be available only with CMAF ingest. We advise you to leverage CMAF Ingest if you can.

6. (Optional) If you're using CMAF inputs, make your selections for **Media quality confidence score (MQCS) settings**.
 - To allow MediaPackage to switch inputs based on the quality score from AWS Elemental MediaLive, choose **Enable input switch based on MQCS**.
 - To include the MQCS in the outputs from your endpoints on this channel, choose **Enable MQCS publishing in Common Media Server Data (CMSD)**.
 - (Optional) To specify which input MediaPackage should prefer when both inputs have equal MQCS scores, choose a value for **Preferred input**. Select **Input 1** to prefer the first ingest endpoint, or **Input 2** to prefer the second ingest endpoint. If you don't specify a preferred input, MediaPackage uses its default switching behavior when MQCS scores are equal.

The preferred input setting helps ensure consistent behavior across multiple MediaPackage channels in different regions when using cross-region architectures.

You can choose one, both, or none of the settings.

For information about MQCS, including workflow set up requirements and how preferred input selection works, see [Leveraging media quality scores with AWS Elemental MediaPackage](#). For information about CMSD headers, see [CMSD headers from AWS Elemental MediaPackage](#).

7. (Optional) If you're using CMAF inputs, choose the **Output locking mode** for your channel:

- **Epoch locked** (default) – Synchronizes segment boundaries across channels for cross-region failover. Use this mode when you need redundant channels in multiple AWS Regions to produce synchronized manifests.
- **Non-epoch locked** – Decouples segment alignment from epoch references. Use this mode for single-region workflows where you want the HLS target duration to be the same as the configured output segment duration.

 Important

You can't change the output locking mode after you create the channel. Non-epoch-locked channels don't support cross-region synchronization or failover.

For more information about output locking modes, see [Output locking mode for CMAF channels](#).

8. Choose your channel's IAM policy settings from the following options:

- **Don't attach a policy** – Restrict access to only those who have access to this account's credentials.

Choose this option if you're using AWS Elemental MediaLive in the same account as MediaPackage.

- **Attach a custom policy** – Define your own policy and restrict access to as few or as many accounts and resources you want. Enter a valid JSON object with the same structure as other IAM policies. The policy should follow the standard security advice of granting least privilege, or granting only the permissions required to perform a task.

If you're not using MediaLive, choose this option and define your policy.

For more information about policies, see [Resource-based policy examples](#).

9. Choose **Create**.

MediaPackage displays the new channel's details page.

After you create a channel, MediaPackage provides two ingest endpoint domain URLs that are fixed for the lifetime of the channel. The channel is active and can start receiving content

as soon as it's created. Provide this information for the upstream encoder stream destination settings. MediaPackage dynamically adjusts resources to increase capacity for your traffic.

If you're using input redundancy and one of the inputs stops sending content, then MediaPackage automatically switches to the other input for the source content. For more information about how input redundancy works, see [Live input redundancy AWS Elemental MediaPackage processing flow](#).

While you create a channel, if you exceed the quotas on the account, you'll receive an error. The error will be similar to Too many requests, please try again. Resource limit exceeded. This error means that either you exceeded the API request quotas, or that you reached the maximum number of channels that your account permits.

To permit downstream video players and content delivery networks (CDNs) to request content playback from MediaPackage, you must add an origin endpoint to a channel.

For instructions on adding endpoints to a channel from the MediaPackage console, see [the section called "Working with endpoints"](#).

Viewing channel details in AWS Elemental MediaPackage

These steps show how to view all channels that are configured in AWS Elemental MediaPackage. You can view specific channel details, including the origin endpoints that are associated with it. You can use the MediaPackage console, the AWS Command Line Interface (AWS CLI), or the MediaPackage API to view channel details.

To view a channel

1. Access the channel group that the channel is associated with, as described in [Viewing channel group details in AWS Elemental MediaPackage](#).

The console shows all existing channels that are configured in MediaPackage.

2. (Optional) To adjust your viewing preferences, choose **Preferences**. For example, you can adjust the page size and properties that you want to view.
3. To view more information about a specific channel, select that channel from the **Channels** list.

MediaPackage displays important information such as the values for ingest endpoint domain URLs, ARN, and the channel policy.

Editing a channel in AWS Elemental MediaPackage

These steps show how to edit the description on a channel in MediaPackage and your channel's policy settings. You can't edit the name of the channel.

You can use the MediaPackage console, MediaPackage API, or AWS CLI to edit a channel. When you're editing a channel, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn't require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

To edit a channel

1. Access the channel group that the channel is associated with, as described in [Viewing channel group details in AWS Elemental MediaPackage](#).
2. To edit a specific channel, select that channel from the **Channels** list.
3. On the channel's details page, choose **Edit**.
4. Make the changes that you want.
5. Choose **Update**.

Resetting channel history in AWS Elemental MediaPackage

These steps show how to reset a channel in MediaPackage. Resetting the channel history clears out previously ingested content from the channel. For information about when you might want to reset, see [Reset for AWS Elemental MediaPackage channels and endpoints](#).

You can use the MediaPackage console, MediaPackage API, or AWS CLI to edit a channel. This guide shows how to reset channel history using the MediaPackage console.

To reset a channel (console)

1. Stop the encoder. If you don't stop the encoder, all endpoints for the MediaPackage channel will stop working.

In AWS Elemental MediaLive, stop the channel as described in [Starting, stopping, and pausing a channel](#) in the MediaLive user guide.

2. In MediaPackage, access the channel group that the channel is associated with, as described in [Viewing channel group details in AWS Elemental MediaPackage](#).
3. From the **Channels** list, select the channel that you want to reset and choose **Reset history**.
4. Wait at least 30 seconds after MediaPackage channel reset to complete, then restart the encoder.

Refreshed content will then be available from the channel's endpoint.

Deleting a channel in AWS Elemental MediaPackage

These steps show how to delete a channel to stop AWS Elemental MediaPackage from receiving further content. Before you can delete the channel, you must delete the channel's origin endpoints as described in [Deleting an endpoint in AWS Elemental MediaPackage](#). You can use the MediaPackage console, the AWS CLI, or the MediaPackage API to delete a channel.

To delete a channel

1. Access the channel group that the channel is associated with, as described in [Viewing channel group details in AWS Elemental MediaPackage](#).
2. Select the name of the channel that you want to delete.
3. Choose **Delete**.
4. Choose **Delete** in the confirmation dialog box.

Working with origin endpoints in AWS Elemental MediaPackage

An origin endpoint is part of a channel and represents the packaging aspect of MediaPackage. When you create an endpoint on a channel, you indicate what streaming format, packaging parameters, and features the output stream will use. Downstream devices request content from the endpoint. Direct your CDNs to the channel group egress domain for stream delivery from MediaPackage. A channel can have multiple endpoints.

Additionally, the endpoint holds information about digital rights management (DRM) and encryption integration, stream bitrate presentation order, and more.

Topics

- [Creating an origin endpoint in AWS Elemental MediaPackage](#)
- [Viewing an origin endpoint in AWS Elemental MediaPackage](#)
- [Editing an endpoint in AWS Elemental MediaPackage](#)
- [Resetting an endpoint in AWS Elemental MediaPackage](#)
- [Deleting an endpoint in AWS Elemental MediaPackage](#)
- [Previewing a manifest from AWS Elemental MediaPackage](#)

Creating an origin endpoint in AWS Elemental MediaPackage

These steps show how to create an origin endpoint (endpoint) on a channel to define how MediaPackage prepares content for delivery. Content can't be served from a channel until it has an endpoint. If you're using input redundancy, each endpoint receives content from one ingest URL at a time. If MediaPackage performs a failover on the inputs for one ingest input URL, the endpoints automatically start receiving content from the other ingest URL. For more information about input redundancy and failover, see [Live input redundancy AWS Elemental MediaPackage processing flow](#).

You can use the MediaPackage console, MediaPackage API, or AWS CLI to create an origin endpoint. When you're creating an origin endpoint, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn't require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

Dynamic Multiview restrictions

If this endpoint is on a Dynamic Multiview channel, additional restrictions apply to its manifests, playback features, and supported output formats. For example, low-latency HLS, Microsoft Smooth Streaming, harvest jobs and live-to-VOD, and time-shifted playback aren't supported on multiview endpoints. For the full list, see [Constraints](#) in [Dynamic Multiview](#).

To create an endpoint

1. Access the channel that the endpoint will be associated with, as described in [Viewing channel details in AWS Elemental MediaPackage](#).

2. Choose **Create endpoint** from the **Origin endpoints** list.
3. Complete the fields as described in the following topics:
 - [Endpoint settings fields](#)
 - [Segment settings fields](#)
 - [Encryption fields](#)
 - [Endpoint policy fields](#)
 - [Manifest fields](#)
4. Choose **Create**.

When you're creating an endpoint, you will receive an error if you exceed the quotas on the account. An error similar to Too many requests, please try again. Resource limit exceeded means that either you've exceeded the API request quotas, or you've already reached the maximum number of endpoints permitted on this channel.

Endpoint settings fields

The endpoint settings fields hold general information about the endpoint.

1. For **Name**, enter a name that describes the origin endpoint. This is the name that you use for API and console interactions. The name is the primary identifier for the endpoint and must be unique for your account in the AWS Region and channel. Supported characters are **A-Z**, **a-z**, **0-9**, **_** (underscore), and **-** (hyphen) with a length of 1 to 256 characters. You can't use spaces in the name, and you can't change the name after you create the endpoint.
2. (Optional) For **Description**, enter any descriptive text that helps you to identify the origin endpoint.
3. For **Container type**, choose the type of container to attach to this origin endpoint. The container type you choose impacts the segment settings, encryption methods, and manifests you can choose.

The container type options are:

- TS (available for HLS and LL-HLS manifests)
- CMAF (available for HLS, LL-HLS, and DASH manifests)
- ISM (available for MSS manifests only. Note: MSS only supports PlayReady DRM, does not support SCTE-35, key rotation, or constant IV)

4. For **Startover window (sec.)**, enter the size of the window (in seconds) to create a window of the live stream that's available for on-demand viewing. Viewers can start-over or catch-up on content that falls within the window. The maximum startover window is 1,209,600 seconds (14 days). For more information about implementing start-over and catch-up TV, see [Time-shifted viewing](#).

You must define a startover window if you're using some settings in the **Filter configuration** fields in the [Manifest fields](#).

5. For **Force endpoint error configuration**, choose the types of problematic situations where you want MediaPackage to return a 404 response on manifests and segments requests:
 - **Stale Manifests:** MediaPackage stops receiving ingest streams on its input, indicating the encoder or network path have failed. This results in output stale manifests that don't get updated any more, which results in playback sessions stopping.
 - **Incomplete Manifests:** MediaPackage is receiving ingest segments, but there are gaps in the timeline. This results in manifests presenting an incomplete timeline for some renditions, potentially generating playback problems.
 - **Missing DRM Keys:** MediaPackage was unable to retrieve an encryption key on a key-rotation operation. This results in the old encryption key still being used after the key-rotation time. While it will not affect playback, it could be problematic from a business or content-rights-entitlement perspective.
 - **Slate Input:** MediaPackage detects that the ingest stream(s) are flagged as including a significant proportion of slate contents (black video frames, audio silence). While playback continues, no content will be displayed by the players, which is a valid reason to failover to a different MediaPackage origin.

When MediaPackage is configured to react to these problems, it will return 404 responses until it can present a consistent timeline in the manifests - meaning that all problematic segments will need to be evicted from the exposed DVR window in the manifest.

Important

Triggering such forced 404 responses is an optional behavior that should be used only if you want to implement CDN-driven failover between multiple MediaPackage origins. For more information on this types of workflow, see [Cross-Region failover](#).

6. For **URI separator**, choose the separator character that MediaPackage uses in generated child manifest and segment URIs. Choose from the following options:

- **Underscore** – Uses an underscore (`_`) character as the separator in URIs. This is the default setting.
- **Hyphen** – Uses a hyphen (`-`) character as the separator in URIs.

This setting applies to all manifest types on the endpoint.

7. For **Stream name output mode**, choose how MediaPackage names streams in egress manifests. This setting is only available for channels that use HLS input type. You can't change this setting after you create the endpoint. Choose from the following options:

- **Index** – Uses numeric indices for stream names (for example, 1, 2, 3). This is the default setting.
- **Passthrough name** – Uses the stream names from the input manifest. For muxed input, MediaPackage automatically disambiguates audio and subtitle renditions by appending the track type and language to the stream name.

Segment settings fields

The segment settings fields hold general information about the segment.

1. For **Segment name**, enter a name that describes the segment. The name is the base name of the segment used in all content manifests inside of the endpoint. Supported characters are **A-Z**, **a-z**, **0-9**, **_** (underscore), and **-** (hyphen) with a length of 1 to 256 characters. You can't use spaces in the name.
2. For **Segment duration (sec.)**, enter the duration (in seconds) of each segment. Enter a value equal to, or a multiple of, the input segment duration. The maximum segment duration is 30 seconds. If the value that you enter is different from the input segment duration, MediaPackage rounds segments to the nearest multiple of the input segment duration.
3. (Optional) For non-epoch-locked CMAF channels, choose the **Output timestamp mode** to control how presentation timestamp (PTS) values appear in output segments:
 - **Passthrough** (default) – Preserves the raw PTS values from the input stream without modification.
 - **Rebased to channel start** – Rebases output PTS values to start close to 0 relative to the first input segment, producing smaller PTS values in output segments.

 Important

You can't change the output timestamp mode after you create the origin endpoint. This setting is only available on non-epoch-locked CMAF channels.

For more information about output timestamp modes, see [Output timestamp mode for non-epoch-locked channels](#).

4. Select **Include IFrame-only stream** to include an additional I-frame only stream along with the other tracks in the manifest. MediaPackage generates an I-frame only stream from the first rendition in the manifest. The service inserts EXT-I-FRAMES-ONLY tags in the output manifest, and then generates and includes an I-frames only playlist in the stream. This playlist enables player functionality like fast forward and rewind.
5. For TS containers only, select **Use audio rendition group** to group all audio tracks into a single rendition group. All other tracks in the stream can be used with any audio rendition from the group. For more information about rendition groups, see [AWS Elemental MediaPackage rendition groups reference](#).
6. For TS containers only, select **Include DVB subtitles** to pass through digital video broadcasting (DVB) subtitles into the output. By default, MediaPackage excludes all DVB subtitles from the output.
7. Select **Enable SCTE support** to include SCTE configuration options. If you select this, you can further define your SCTE configuration in additional fields.
8. For **SCTE filtering**, choose the SCTE-35 message types that will be ad markers in the output. If you don't make a selection here, by default, MediaPackage inserts all ad markers in the output manifest.
 - Splice insert
 - Break
 - Provider advertisement
 - Distributor advertisement
 - Provider placement opportunity
 - Distributor placement opportunity
 - Provider overlay placement opportunity
 - Distributor overlay placement opportunity

- Program
 - Chapter
 - Unscheduled event
 - Alternate content opportunity
 - Network
 - Provider promo
 - Distributor promo
 - Provider ad block
 - Distributor ad block
9. For **SCTE in segments**, choose whether to include SCTE-35 messages in the segment files. Choose from the following options:
- **None** – SCTE-35 messages are not included in segment output. This is the default setting.
 - **All** – All SCTE-35 messages are included in the segment data. For DASH manifests, an `InbandEventStream` tag is added to signal the presence of SCTE messages.
 - **Matches filter** – Only SCTE-35 messages from events whose type matches the configured SCTE filtering are included in the segment data. For DASH manifests, an `InbandEventStream` tag is added to signal the presence of SCTE messages.
10. For **Custom ad types**, choose additional SCTE-35 event types that you want MediaPackage to treat as advertisements. By default, MediaPackage treats only splice insert and time signal placement opportunity events as ads. Use this setting to designate other event types as ads. Choose from the following values:
- PROGRAM
 - CHAPTER
 - UNSCHEDULED_EVENT
 - ALTERNATE_CONTENT_OPPORTUNITY
 - NETWORK

Encryption fields

Protect your content from unauthorized use through content encryption and digital rights management (DRM). MediaPackage uses the [AWS Secure Packager and Encoder Key Exchange \(SPEKE\) API](#) to facilitate content encryption and decryption by a DRM provider. Using SPEKE, the DRM provider supplies encryption keys to MediaPackage through the SPEKE API. The DRM provider

also supplies licenses to supported media players for decryption. For more information about how SPEKE is used with services and features running in the cloud, see [AWS cloud-based architecture](#) in the *Secure Packager and Encoder Key Exchange API Specification guide*.

 Note

To encrypt content, you must have a DRM provider, and be set up to use encryption. For information, see [the section called "Content encryption and DRM"](#).

1. Choose **Encrypt content** to serve content with copyright protection.
2. For **Encryption method**, choose the encryption method to use. If you don't see your preferred encryption method, confirm you choose the correct container type. The encryption method you choose impacts the DRM system providers you can choose. For supported encryption methods and DRM system providers, see [Container and DRM system support with SPEKE](#).
 - The valid encryption methods for TS container types are:
 - AES-128
 - Sample AES
 - The valid encryption methods for CMAF container types are:
 - CENC
 - CBCS
 - The valid encryption method for ISM container types is:
 - CENC
3. For **DRM systems**, choose the DRM system providers you're using to protect your content during distribution. You can choose more than one. If you don't see your DRM system provider, confirm you choose the correct container type and encryption method. For supported DRM system providers, see [Container and DRM system support with SPEKE](#).

The valid DRM systems for TS container types are:

- Fairplay (Sample AES only)
- Clear Key AES-128 (AES-128 only)

The valid DRM systems for CMAF container types are:

- PlayReady
- Widevine

- Irdeto (CENC only)
- Fairplay (CBCS only)

The valid DRM system for ISM container types is:

- PlayReady
4. For **Resource ID**, enter an identifier for the content. The service sends this to the key server to identify the current endpoint. How unique you make this depends on how fine-grained you want access controls to be. The service does not permit you to use the same ID for two simultaneous encryption processes. The resource ID is also known as the content ID.

The following example shows a resource ID.

```
MovieNight20171126093045
```

5. For **Key server URL**, enter the URL of the API Gateway proxy that you set up to talk to your key server. The API Gateway proxy must reside in the same AWS Region as MediaPackage.

The following example shows a URL.

```
https://1wm2dx1f33.execute-api.us-west-2.amazonaws.com/SpekeSample/copyProtection
```

6. For **Role ARN**, enter the Amazon Resource Name (ARN) of the IAM role that provides you access to send your requests through API Gateway. Get this from your DRM solution provider.

The following example shows a role ARN.

```
arn:aws:iam::accountID:role/SpekeAccess
```

7. (Optional) For **Certificate ARN**, enter the Amazon Resource Name (ARN) of a certificate stored in ACM for content key encryption.

The following example shows a certificate ARN.

```
arn:aws:acm:Region:accountID:certificate/certificateID
```

For information about content key encryption, see [Preparing and managing certificates for content key encryption](#).

8. For **Video encryption preset** and **Audio encryption preset**, select the preset for encrypting audio and video. For information about presets, see [Encryption presets in AWS Elemental MediaPackage](#).
9. (Optional) For **Constant initialization vector** enter a 128-bit, 16-byte hex value represented by a 32-character string, used in conjunction with the key for encrypting content. If you don't specify a value, then MediaPackage creates the constant initialization vector (IV).
10. For **Key rotation interval (sec.)**, enter the frequency (in seconds) of key changes for live workflows, in which content is streamed real time. The service retrieves content keys before the live content begins streaming, and then retrieves them as needed over the lifetime of the workflow. By default, key rotation is 300 seconds (5 minutes), the minimum rotation interval, which is equivalent to setting it to 300. The maximum key rotation interval is 31,536,000 seconds (1 year). If you don't enter an interval, content keys aren't rotated.

The following example setting causes the service to rotate keys every thirty minutes.

1800

For information about key rotation, see [AWS Elemental MediaPackage key rotation behavior](#).

11. (Optional) Select **Exclude segment DRM metadata** to omit SEIG and SGPD boxes from CMAF segments. This can improve compatibility with certain devices and players that don't support these DRM metadata boxes.

Note

This option is only available for CMAF container formats. For TS containers, this option is disabled because TS containers don't include SEIG and SGPD boxes.

When enabled, MediaPackage excludes segment-level DRM metadata boxes while preserving other essential DRM functionality. Key rotation can still be handled through media playlist signaling. For more information about DRM segment metadata, see [Managing DRM segment metadata in AWS Elemental MediaPackage](#).

Endpoint policy fields

You must assign a channel policy to enable content to flow into your channel from sources outside of your account.

1. Under **Endpoint policy**, choose an endpoint policy to enable content to flow into your channel from sources outside of your account. For more information about policies, see [Resource-based policy examples](#).
 - **Don't attach a policy** - Restrict access to only those who have access to this account's credentials.
 - **Attach a custom policy** - Define your own policy and restrict access to as few or as many as you want. Enter a valid JSON object with the same structure as other IAM policies. The policy should follow the standard security advice of granting least privilege, or granting only the permissions required to perform a task.
 - **Attach a public policy** - Accept all incoming client requests to a channel's output. Enter a valid JSON object with the same structure as other IAM policies.

Manifest fields

The manifests fields hold general information about the manifest. You must attach at least one manifest to an origin endpoint. You can attach up to twenty five manifests to a single origin endpoint, and request a quota increase if necessary.

Choose the type of manifest to use. You can choose an HLS manifest, LL-HLS manifest, DASH manifest, or MSS manifest. The available manifest types depend on the container type you selected:

- TS container: HLS and LL-HLS manifests
- CMAF container: HLS, LL-HLS, and DASH manifests
- ISM container: MSS manifests only

Note

Microsoft Smooth Streaming (MSS) is primarily used for legacy devices such as older smart TVs, Xbox consoles, and platforms that use Microsoft Silverlight. If you need to support these devices, select the ISM container type and create an MSS manifest. For more information about MSS, see [MSS in AWS Elemental MediaPackage](#).

Create an HLS or LL-HLS manifest

To create an HLS or LL-HLS manifest

1. For **Manifest name** enter a short string that will be appended to the endpoint URL. The manifest name creates a unique path to this endpoint. If you don't enter a value, MediaPackage uses the default manifest name, *index*. MediaPackage automatically inserts the format extension, such as *.m3u8*. Supported characters are **A-Z, a-z, 0-9**, and **-** (hyphen). You can't use underscores in the name.
2. (Optional) For **Child manifest name** enter a short string that will be appended to the endpoint URL. The child manifest name creates a unique path to this endpoint. Supported characters are **A-Z, a-z, 0-9**, and **-** (hyphen). You can't use underscores in the name.
3. For **Manifest window (sec.)** enter the total duration (in seconds) of the manifest's content. The maximum manifest window is 900 seconds (15 minutes).
4. For **Program date/time interval (sec.)** enter the interval (in seconds) for MediaPackage to insert the EXT-X-PROGRAM-DATE-TIME tags in the manifest. Program date time (PDT) is optional when using HLS manifests, but is required when using low-latency HLS manifests.

The maximum PDT interval is 1,209,600 seconds (14 days). If you don't enter an interval, EXT-X-PROGRAM-DATE-TIME tags aren't included in the manifest.

The EXT-X-PROGRAM-DATE-TIME tag holds the time of the segment. When PDT information is available in the source content, MediaPackage uses this same information on the output content. Otherwise, MediaPackage uses Coordinated Universal Time (UTC) for the PDT.

The PDT information helps downstream players to synchronize the stream to the wall clock, enabling functionality like viewer seek in the playback timeline and time display on the player.

5. Select **URL-encode HLS child manifest query parameters** if you need query parameters to be encoded to comply with AWS Signature Version 4 signing. For more information about this setting, see [URL encoding query parameters](#).
6. Choose **Enable start tag** to insert an EXT-X-START tag in your manifest. Set the offset for when the tag appears, and indicate if this is a precise offset. MediaPackage adds EXT-X-START tags to the output manifest based on your specified offset settings.
 - If the offset is *positive*, it must be less than the configured manifest duration, minus three times the configured segment target duration (in accordance with the [HLS specification](#)).
 - If the offset is *negative*, the absolute value must be three times the configured segment target duration and it must be smaller than the configured manifest duration.

7. If you chose **Enable SCTE support**, for **Ad markers**, choose how ad markers are included in the packaged content. If you include ad markers in the content stream in your upstream encoders, then you need to inform MediaPackage what to do with the ad markers in the output. If you don't see this field, select **Enable SCTE support** in the origin endpoint segment settings. Choose from the following options:
 - **Daterange** – Insert EXT-X-DATERANGE tags to signal ads and program transition events in TS and CMAF output manifests. If you choose daterange, you *must* also enter a **Program date/time interval (sec.)** value of 1 or greater.
 - **SCTE-35 enhanced** – Generate industry-standard CUE tag ad markers in HLS manifests based on SCTE-35 input messages from the input stream. This option inserts EXT-X-CUE-OUT, EXT-X-CUE-IN, and related CUE tags to signal ads and program transition events, providing the same SCTE-35 information as DATERANGE tags but in a format compatible with legacy ad insertion systems.
8. For **SCTE in manifests**, choose which SCTE-35 events appear in the output manifests. Choose from the following options:
 - **All** – All SCTE-35 events appear in the manifest. This is the default setting.
 - **Matches filter** – Only SCTE-35 events whose type matches the configured SCTE filtering appear in the manifest. You must have SCTE support enabled with at least one SCTE filter value configured to use this option.
9. For **URI path type**, choose how MediaPackage generates child manifest and segment URIs. Choose from the following options:
 - **Leaf** – URIs are relative paths (leaf). This is the default setting.
 - **Root** – URIs are absolute paths rooted at the endpoint base egress URI.
10. Select **Enable filter configuration** if you want to optionally add filters and settings to modify manifests. These filters apply to all manifests that originate from this endpoint.

To automatically fill these values from an existing query string, choose **Import from query string**. For example, this string provides the following filters:

```
aws.manifestfilter=video_codec:h265;audio_language:fr,en-US,de&start=2023-10-20T12:20:50Z&end=2023-10-20T13:20:50Z&time_delay=10
```

Filter key video_codec, **Filter value** h265, **Filter key** audio_language, **Filter value** fr,en-US, **Start time** 2023-10-20T12:20:50Z, **End time** 2023-10-20T13:20:50Z, and **Time delay** 10 seconds.

Note

When you include a Manifest filter, you cannot use matching query parameters for the manifest's endpoint URL. If you do, you will receive a 404 HTTP error code instead. For example, if you include a Manifest filter with a `audio_sample_rate` Filter key and `44100` Filter value, and you make an HTTP request for `https://<example-url>/?aws.manifestfilter=audio_sample_rate:44100`, you will receive a 404 error.

Manifest filter

Optionally specify one or more manifest filters for all of your manifest egress requests.

You enter a **Filter key** and **Filter value** pair for each manifest filter. Your filter values can be a range, single value, or combination of both.

For a list of supported keys and values, see [Manifest filtering query parameters](#).

Start time and End time

Optionally specify the start or end time for all of your manifest egress requests. To use this setting, you must have a startover window defined.

For more information about start and end times, see [Start and end parameters](#).

If you enter a start or end time using the API, or import using **Import from query string** in the MediaPackage console, enter dates in an ISO-8601 format.

Time delay

Optionally specify the time delay (in seconds) for all of your manifest egress requests. To use this setting, you must have a startover window defined.

For more information about using time delays, see [Time delay](#).

DRM settings

Optionally specify DRM-related settings for HLS manifests. Currently supports excluding session keys from the multivariant playlist.

To exclude session keys, either enter `aws.drmsettings=exclude_session_keys` in the manifest filters or click **Exclude session keys**. This removes EXT-X-SESSION-KEY tags from the multivariant playlist while preserving key information in individual media playlists. This setting is useful for legacy clients that have issues with session key pre-fetching or when using manifest filtering to control access to specific content variants.

This setting only applies to HLS manifests and is ignored for DASH manifests.

Clip start time data and time

When using a time delay, optionally specify the clip start time for all of your manifest egress requests. This option specifies the earliest content that can be included in the manifest. Content ingested before the clip start time will not be included in the manifest. Cannot be used with start time or end time filters. To use this setting, you must have a startover window defined.

For more information about clip start and time delay, see [Using clip start with time delay](#).

Important

If you're using this setting to limit content playback for contractual reasons, this parameter should be set at your CDN, and not at the client.

Create a DASH manifest

Use the following steps to configure the endpoint to create manifests that are compliant with DASH.

DVB-DASH requirements

If the outputs from this endpoint will be compliant with DVB-DASH, you must adhere to the following requirements:

- The channel must use CMAF input
- The endpoint must not use **UTC Direct** for UTC timing mode

To create a DASH manifest

1. For **Manifest name**, enter a short string that will be appended to the endpoint URL. The manifest name creates a unique path to this endpoint. If you don't enter a value, MediaPackage uses the default manifest name, *index*. MediaPackage automatically inserts the format extension, such as .mpd. Supported characters are **A-Z**, **a-z**, **0-9**, and **-** (hyphen). You can't use underscores in the name.
2. For **Manifest window (sec.)** enter the total duration (in seconds) of the manifest's content. The maximum manifest window is 900 seconds (15 minutes). You can request a quota increase if necessary.
3. For **Min update period (sec.)**, enter the minimum amount of time (in seconds) that the player should wait before requesting manifest updates. A lower value means that manifests are updated more frequently, but a lower value also contributes to request and response network traffic.
4. For **Min buffer time (sec.)**, enter the minimum amount of time (in seconds) that a player must keep in the buffer. If network conditions interrupt playback, the player will have additional buffered content before playback fails, allowing for recovery time before the viewer's experience is affected.
5. For **Suggested presentation delay (sec.)**, enter the amount of time (in seconds) that the player should be from the end of the manifest. This sets the content start point back x seconds from the end of the manifest (the point where content is live). For example, with a 35-second presentation delay, requests at 5:30 receive content from 5:29:25. When used with time delay, MediaPackage adds the suggested presentation delay to the time delay duration.
6. For **Segment template format**, choose how MediaPackage and playback requests refer to each segment.
 - For **Number with timeline**, MediaPackage uses the `$Number$` variable to refer to the segment in the `media` attribute of the `SegmentTemplate` tag. The value of the variable is the sequential number of the segment. `SegmentTimeline` is included in each segment template.
7. For **Manifest compactness**, indicate if you want MediaPackage to serve a standard (compacted) or full manifest (no compacting) in response to playback requests.
 - If you choose **None**, MediaPackage presents the `SegmentTemplate` and `SegmentTimeline` tags for every `Representation` in the manifest.
 - **Standard** is the default selection. It indicates that MediaPackage combines duplicate `SegmentTemplate` tags and presents them at the start of the manifest. This shortens the manifest and makes it easier for some devices to process it.

For more information about the manifest layout options, see [DASH manifest compactness](#).

8. For **Audio timeline pattern**, choose how MediaPackage represents audio segment durations in the SegmentTimeline.

- If you choose **None**, MediaPackage lists each audio segment duration individually in the SegmentTimeline. This is the default setting.
- **Patterned** enables Segment Duration Patternization (SDP). MediaPackage detects repeating cycles in audio segment durations and represents them as Pattern elements, reducing manifest size for long time-shift buffers.

For more information about audio timeline patterns, see [DASH audio timeline pattern](#).

9. For **Profiles**, optionally choose if the output is compliant with DVB-DASH. Review the DVB-DASH requirements to ensure the endpoint is compliant with this profile:

- The channel must use CMAF input
- The endpoint must not use **UTC Direct** for UTC timing mode

10 For **UTC timing**, select the method that the player uses to synchronize to coordinated universal time (UTC) wall clock time. This enables the player and MediaPackage to run on the same UTC wall clock time. This is a requirement, otherwise playback timing or synchronization issues can occur. Choose from the following options:

- UTC Direct
- HTTP Head
- HTTP Iso
- HTTP Xsdate

11 For **UTC timing source**, specify a URI to use for UTC synchronization. This is the URI used to fetch the timing data according to the scheme defined by **UTC timing**. This value is only valid if **UTC timing** is not NONE or UTC DIRECT. This value will be set as the @value attribute for the UTCTiming element. For information about @value, see [DASH clock synchronization](#).

12 (Optional) For **Availability start time**, enter a custom timestamp to use as the availabilityStartTime attribute in the DASH MPD. MediaPackage calculates the Period@start value relative to this timestamp. You can use this setting to reduce large Period@start values.

If you don't specify a value, MediaPackage uses the default availability start time of 2024-01-01T00:00:00Z. You configure this setting for each manifest, so each DASH manifest on the endpoint can use a different value.

The timestamp must meet the following requirements:

- Must have hourly granularity only (minutes, seconds, and fractional seconds must be zero)
- Must be on or after 2024-01-01T00:00:00Z
- Must be at least 14 days in the past

13For **DASH period triggers**, choose how MediaPackage creates media presentation description (MPD) periods in the DASH output manifest. For more information, see [Multi-period DASH in AWS Elemental MediaPackage](#). Choose from the following options:

- **Avails** – Avails that pass the `ScteFilter` will create new periods.
- **DRM key rotation** – Encryption key rotation will create new periods.
- **Source changes** – Changes in the stream set will create new periods.
- **Source disruptions** – Gaps in all content streams will create new periods.
- **None** – MediaPackage formats the manifest as a single period. It doesn't create additional periods, unless DRM settings change.

14Select **Configure subtitle TTML profile** to optionally configure the profile that TTML subtitles use.

Subtitle TTML profile

The profile that MediaPackage uses when signaling subtitles in the manifest. **IMSC** is the default profile. **EBU-TT-D** produces subtitles that are compliant with the EBU-TT-D TTML profile. MediaPackage passes through subtitle styles to the manifest. For more information about EBU-TT-D subtitles, see [EBU-TT-D Subtitling Distribution Format](#).

15Select **Configure program information** to optionally provide details about the content that you want MediaPackage to pass through in the manifest to the playback device. To read more about program information, see the *Program information* section of the [ISO 23009-1 DASH standards](#).

Title

The title for the manifest.

Source

Information about the content provider.

Copyright

A copyright statement about the content.

Language code

The language code for this manifest.

More information URL

An absolute URL that contains more information about this content.

16Select **Configure DASH base URLs** to optionally specify one or more locations for the segments. For more information, see the *Base URL* section of the [ISO 23009-1 DASH standards](#) and *Handling of BaseURLs by players* in the DVB.org [MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP Based Networks document](#).

URL

The Base URL.

Service location

The name of the Base URL location.

DVB priority

For endpoints that use the DVB-DASH profile only. The priority that the playback device should use this Base URL. Lower values are higher priority.

DVB weight

For endpoints that use the DVB-DASH profile only. The weighting for Base URLs with the same priority.

17For endpoints that use the DVB-DASH profile, select **Configure DVB-DASH settings** to optionally pass to the player information for downloading subtitle fonts and sending error reports to pass through in the manifest to the playback device. For more information about these settings, see *DVB font download scheme* and *DVB metrics reporting* in the DVB.org [MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP Based Networks document](#).

These settings are compatible only with the DASH-DVB profile. **Profiles** must be set to **DVB-DASH** to set these fields.

Font download URL

The URL of the font to download for subtitles.

Font family

The `fontFamily` name for subtitles, as described in [EBU-TT-D Subtitling Distribution Format](#).

MIME type

The `mimeType` of the resource that's at the indicated **Font download URL**. Options are **application/font-sfnt** and **application/font-woff**, as described in the [DVB.org DASH document](#) linked above.

Error metrics reporting URL

The URL to which a reporting player will send error reports.

Probability

The number of playback devices per 1000 that will send error reports to the reporting URL. This integer represents the probability that the playback device will be a reporting player for this session.

18For **Ad markers**, choose how ad markers are signaled in the output manifests. All the non-ad markers that you include in the content stream in your upstream encoders will also be present in the output manifests. Choose from the following options:

- **Binary** – The SCTE-35 marker is expressed as a hex-string (Base64 string) rather than full XML.
- **XML** – The SCTE marker is expressed fully in XML.

19For **SCTE in manifests**, choose which SCTE-35 events appear in the output manifests. Choose from the following options:

- **All** – All SCTE-35 events appear in the manifest. This is the default setting.
- **Matches filter** – Only SCTE-35 events whose type matches the configured SCTE filtering appear in the manifest. You must have SCTE support enabled with at least one SCTE filter value configured to use this option.

20For **URI path type**, choose how MediaPackage generates child manifest and segment URIs. Choose from the following options:

- **Leaf** – URIs are relative paths (leaf). This is the default setting.
- **Root** – URIs are absolute paths rooted at the endpoint base egress URI.

21Select **Enable filter configuration** if you want to optionally add filters and settings to modify manifests. These filters apply to all manifests that originate from this endpoint.

To automatically fill these values from an existing query string, choose **Import from query string**. For example, this string provides the following filters:

```
aws.manifestfilter=video_codec:h265;audio_language:fr,en-US,de&start=2023-10-20T12:20:50Z&end=2023-10-20T13:20:50Z&time_delay=10
```

Filter key video_codec, **Filter value** h265, **Filter key** audio_language, **Filter value** fr,en-US, **Start time** 2023-10-20T12:20:50Z, **End time** 2023-10-20T13:20:50Z, and **Time delay** 10 seconds.

Note

When you include a Manifest filter, you cannot use matching query parameters for the manifest's endpoint URL. If you do, you will receive a 404 HTTP error code instead. For example, if you include a Manifest filter with a audio_sample_rate Filter key and 44100 Filter value, and you make an HTTP request for `https://<example-url>/?aws.manifestfilter=audio_sample_rate:44100`, you will receive a 404 error.

Manifest filter

Optionally specify one or more manifest filters for all of your manifest egress requests.

You enter a **Filter key** and **Filter value** pair for each manifest filter. Your filter values can be a range, single value, or combination of both.

For a list of supported keys and values, see [Manifest filtering query parameters](#).

Start time and End time

Optionally specify the start or end time for all of your manifest egress requests. To use this setting, you must have a startover window defined.

For more information about start and end times, see [Start and end parameters](#).

If you enter a start or end time using the API, or import using **Import from query string** in the MediaPackage console, enter dates in an ISO-8601 format.

Time delay

Optionally specify the time delay (in seconds) for all of your manifest egress requests. To use this setting, you must have a startover window defined.

For more information about using time delays, see [Time delay](#).

Clip start time data and time

When using a time delay, optionally specify the clip start time for all of your manifest egress requests. This option specifies the earliest time that the content can result in a manifest. Content ingested before the clip start time will not be included in the manifest. Cannot be used with start time or end time filters. To use this setting, you must have a startover window defined.

For more information about clip start and time delay, see [Using clip start with time delay](#).

Important

If you're using this setting to limit content playback for contractual reasons, this parameter should be set at your CDN, and not at the client.

Create an MSS manifest

This topic shows you how to configure an endpoint in AWS Elemental MediaPackage to create Microsoft Smooth Streaming (MSS) manifests. MSS manifests require the ISM container type.

MSS requirements and limitations

When creating an MSS manifest, note the following requirements and limitations:

Configuration requirements

- You must select ISM as the container type
- Only PlayReady DRM is supported for encryption

Feature limitations

- SCTE-35 is not supported
- Key rotation is not supported for encrypted content

- Lookahead fragments are automatically set to 2 and are not configurable
- CDK constructs support may be limited (contact AWS Support for current status)
- Auto support for ISM constructs is not enabled

Technical specifications

- URLs end with `.ism/Manifest` instead of `.html`
- Manifest MIME type is `text/xml`

For information about URL limits and endpoint quotas, see [Quotas in AWS Elemental MediaPackage](#).

To create an MSS manifest

1. For **Manifest name**, enter a short string that will be appended to the endpoint URL. The manifest name creates a unique path to this endpoint. If you don't enter a value, MediaPackage uses the default manifest name, *index*. MediaPackage automatically inserts the format extension, such as `.ismv`. Supported characters are **A-Z**, **a-z**, **0-9**, and **-** (hyphen). You can't use underscores in the name.
2. For **Manifest window (sec.)** enter the total duration (in seconds) of the manifest's content. The default value is 60 seconds. The maximum manifest window is 900 seconds (15 minutes). You can request a quota increase if necessary.
3. For **Manifest layout**, choose how MediaPackage formats the manifest:
 - **Full** – Each fragment in the sequence has an explicit value for the duration field and an implicit value for the time field, except the first fragment, whose start-time is explicit.
 - **Compact** – Uses the `FragmentRepeat` field on segments that share the same attributes and length to reduce manifest size. This can significantly reduce manifest length for streams with consistent segment durations.

For most use cases, **Compact** is recommended as it reduces manifest size and network overhead.

4. For **Lookahead fragment count**, the value is automatically set to 2 and cannot be changed for MSS manifests. This buffer helps ensure smooth playback.

MSS includes the concept of server lookahead, where a buffer of segments is held back by the server until future segments are available. This means that segments right at the live edge will

not be included in the manifest until the specified number of lookahead fragments are internally available.

The lookahead is ignored when the stream has ended or when there's a discontinuity between segments.

5. Select **Enable filter configuration** if you want to optionally add filters and settings to modify manifests. These filters apply to all manifests that originate from this endpoint.

To automatically fill these values from an existing query string, choose **Import from query string**.

Note

When you include a Manifest filter, you cannot use matching query parameters for the manifest's endpoint URL. If you do, you will receive a 404 HTTP error code instead.

Manifest filter

Optionally specify one or more manifest filters for all of your manifest egress requests.

You enter a **Filter key** and **Filter value** pair for each manifest filter. Your filter values can be a range, single value, or combination of both.

For a list of supported keys and values, see [Manifest filtering query parameters](#).

Start time and End time

Optionally specify the start or end time for all of your manifest egress requests. To use this setting, you must have a startover window defined.

For more information about start and end times, see [Start and end parameters](#).

If you enter a start or end time using the API, or import using **Import from query string** in the MediaPackage console, enter dates in an ISO-8601 format.

Time delay

Optionally specify the time delay (in seconds) for all of your manifest egress requests. To use this setting, you must have a startover window defined.

For more information about using time delays, see [Time delay](#).

Clip start time data and time

When using a time delay, optionally specify the clip start time for all of your manifest egress requests. This option specifies the earliest content that can be included in the manifest. Content ingested before the clip start time will not be included in the manifest. Cannot be used with start time or end time filters. To use this setting, you must have a startover window defined.

For more information about clip start and time delay, see [Using clip start with time delay](#).

Important

If you're using this setting to limit content playback for contractual reasons, this parameter should be set at your CDN, and not at the client.

After you've configured your MSS manifest, choose **Create** to create the endpoint. For information about testing your MSS endpoint, see [Testing MSS playback in AWS Elemental MediaPackage](#). For troubleshooting information, see [Troubleshooting MSS endpoints in AWS Elemental MediaPackage](#).

Viewing an origin endpoint in AWS Elemental MediaPackage

These steps show how to view all origin endpoints that are configured in AWS Elemental MediaPackage. You can view the details about a specific endpoint to obtain its playback URL, the packaging settings, and the manifests within the endpoint. You can use the MediaPackage console, the AWS CLI, or the MediaPackage API to view the details of an endpoint.

To view an origin endpoint

1. Access the channel that the endpoint is associated with, as described in [Viewing channel details in AWS Elemental MediaPackage](#).

The console shows all existing origin endpoints that are configured in MediaPackage.

2. (Optional) To adjust your viewing preferences, choose **Preferences**. For example, you can adjust the page size and properties that you want to view.
3. To view more information about a specific origin endpoint, select that origin endpoint from the **Origin Endpoints** list. For downstream device requests, you must provide the endpoint URL from the **Endpoint URL** field or the CloudFront CDN URL.

Editing an endpoint in AWS Elemental MediaPackage

Edit the packaging preferences on an endpoint in MediaPackage to optimize the viewing experience. You can't change the container type after you save an endpoint or greyed-out fields. To serve content with a different packager, create a different endpoint.

Any edits you make that impact the video output may not be reflected for a few minutes.

You can use the MediaPackage console, MediaPackage API, or AWS CLI to edit an origin endpoint. When you're editing an origin endpoint, don't put sensitive identifying information like customer account numbers into free-form fields such as the name or description field. MediaPackage doesn't require that you supply any customer data. This includes when you work with MediaPackage using the MediaPackage console, MediaPackage API, AWS CLI, or AWS SDKs. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or Amazon CloudWatch Events.

To edit an endpoint

1. Access the channel that the endpoint is associated with, as described in [Viewing channel details in AWS Elemental MediaPackage](#).

The console shows all existing origin endpoints that are configured in MediaPackage.

2. Under **Origin endpoints**, choose the endpoint that you want to edit and then choose **Edit endpoint**.
3. Edit the endpoint options that you want to change.
4. Choose **Edit**.

Resetting an endpoint in AWS Elemental MediaPackage

These steps show how to reset an origin endpoint in MediaPackage. Resetting the endpoint clears previous content from endpoint egress. For information about when you might want to reset, see [Reset for AWS Elemental MediaPackage channels and endpoints](#).

You can use the MediaPackage console, the AWS CLI, or the MediaPackage API to reset an endpoint.

To reset an endpoint (console)

1. Access the channel that the endpoint is associated with, as described in [Viewing channel details in AWS Elemental MediaPackage](#).

The console shows all existing origin endpoints that are configured in MediaPackage.

2. Under **Origin endpoints**, choose the endpoint that you want to reset and then choose **Reset history**.

MediaPackage might return old content from this endpoint in the first 30 seconds after the endpoint reset. For best results, when possible, wait 30 seconds from endpoint reset to send playback requests to this endpoint.

Deleting an endpoint in AWS Elemental MediaPackage

Endpoints in MediaPackage can serve content until they're deleted. These steps show how to delete the endpoint if it should no longer respond to playback requests. You must delete all endpoints from a channel before you can delete the channel.

Warning

If you delete an endpoint, the playback URL stops working.

You can use the MediaPackage console, the AWS CLI, or the MediaPackage API to delete an endpoint.

To delete an endpoint

1. Access the channel that the endpoint is associated with, as described in [Viewing channel details in AWS Elemental MediaPackage](#).

The console shows all existing origin endpoints that are configured in MediaPackage.

2. Under **Origin endpoints**, choose the endpoint that you want to delete.
3. Choose **Delete**.
4. In the **Delete endpoints** confirmation dialog box, choose **Delete**.

Previewing a manifest from AWS Elemental MediaPackage

Preview an endpoint's manifest to ensure that MediaPackage is receiving the content stream and can package it. The preview is helpful for avoiding playback failures after the endpoint is published and for troubleshooting later if there are any playback issues.

You can use the MediaPackage console to preview playback from the endpoint.

To preview an endpoint's playback

1. Access the channel that the endpoint is associated with, as described in [Viewing channel details in AWS Elemental MediaPackage](#).
2. Under **Origin endpoints**, select the endpoint that you want to preview.
3. To preview playback, do one of the following:
 - Choose **Preview** to play content with the embedded player.
 - Choose **QR code** to view and scan the QR code for playback on a compatible device.

Delivering VOD content from AWS Elemental MediaPackage

At this time, MediaPackage v2 doesn't support video on demand (VOD) workflows. If you're looking to support VOD workflows, see the [AWS Elemental MediaPackage v1 User Guide](#).

Creating live-to-VOD assets with MediaPackage

You can use the AWS Elemental MediaPackage v2 live-to-VOD harvester to extract portions of live video streams and save them as Video on Demand (VOD) assets. This feature is useful for creating highlight reels from live sports events, archiving broadcasts, or any other scenario where you need to convert live content into on-demand assets.

The following topics provide more information about live-to-VOD assets in MediaPackage v2.

Topics

- [Requirements](#)
- [What is a harvest job?](#)
- [How live-to-VOD works](#)
- [Creating a harvest job](#)
- [Viewing harvest jobs](#)
- [Canceling a harvest job](#)

Requirements

When creating live-to-VOD assets in MediaPackage v2, keep the following requirements in mind:

Endpoint requirements

- The endpoint must be a MediaPackage v2 endpoint. MediaPackage v1 endpoints are not compatible.
- The endpoint must grant MediaPackage v2 `GetObject` and `HarvestObject` access. For more information, see [MediaPackage L2V Harvester](#).

Live-to-VOD asset requirements

- The start time of the harvest job must be in the past.
- The maximum job duration is 24 hours.
- Any harvested manifest must not include start, end, or clip start time in the manifest filter configuration.

What is a harvest job?

A harvest job represents a request to extract a live-to-VOD asset from an endpoint for a specific timeframe. MediaPackage uses information from the harvest job to determine the start and end times of the asset, and where to store it after the harvest job is complete.

A harvest job runs only once after it's created. MediaPackage keeps a record of the job on your account for reference only. You can't modify or delete a record after you create the harvest job.

Harvest job features

- Multiple manifests can be harvested in a single request.
- Real-time harvesting is supported for ongoing streams.
- Rendition filtering is available through a combination of `ManifestFilter` in `FilterConfiguration` (in `OriginEndpoint` configuration) and `HarvestedManifests` (in `HarvestJob` configuration).
- All included segments across different manifests will be de-duplicated to be moved to your S3 bucket only once. (The segments are shared between different manifests.)

How live-to-VOD works

In the processing flow for live-to-VOD (video on demand) content, MediaPackage v2 extracts a clip of video from a live content stream. MediaPackage saves this clip as a live-to-VOD asset in Amazon S3. You can use the VOD content processing functionality in MediaPackage to deliver the asset to playback devices, or you can use a VOD encoding service that supports HLS or DASH inputs.

The following sections provide an overview of the live-to-VOD processes.

Live-to-VOD Harvest Job

You can schedule a harvest job to take place at a specified timeframe in your live content stream.

The process looks like this:

1. You create a channel group, channel, and origin endpoint in MediaPackage v2 to ingest and package your live stream.
2. You must grant the MediaPackage v2 service principal access to `PutObject` in your S3 bucket policy and `HarvestObject` in your MediaPackage v2 endpoint.

3. You create a harvest job using the `CreateHarvestJob` API operation or the AWS Management Console. This job defines the live-to-VOD asset you want to extract from the live stream.
4. MediaPackage v2 processes the harvest job, extracting the specified timeframe from the live stream.
5. The harvested asset is saved to the S3 bucket that you specified in the harvest job configuration.
6. The live-to-VOD asset is now available in your S3 bucket for further processing or delivery.

Note

When using harvested content with AWS Elemental MediaTailor, be aware that the harvesting process can result in `BANDWIDTH` values that differ from the original live stream. If `BANDWIDTH` values differ by 15% or more, MediaTailor might determine that previously transcoded ads no longer match and require re-transcoding. For more information about this behavior and troubleshooting steps, see [BANDWIDTH variance threshold](#) in the *MediaTailor User Guide*.

Live-to-VOD HarvestObject

You can specify a packaged manifest or segments to harvest from your live content stream. For information about the `HarvestObject` API operation, see [HarvestObject](#).

The process looks like this:

1. You create a channel group, channel, and origin endpoint in MediaPackage v2 to ingest and package your live stream.
2. You must grant the MediaPackage v2 service principal access to `PutObject` in your S3 bucket policy and `HarvestObject` in your MediaPackage v2 endpoint.
3. You harvest a packaged live-to-VOD asset using the `HarvestObject` dataplane API operation. This request specifies the manifest or segments that you want to save and indicates where they should be saved.
4. MediaPackage v2 processes the request and saves the asset to the S3 bucket that you specified.
5. The live-to-VOD asset is now available in your S3 bucket for further processing or delivery.

Creating a harvest job

Create a harvest job to extract a live-to-VOD asset from an encrypted or clear (unencrypted) live HLS or DASH stream.

Important

To run a harvest job and save the live-to-VOD asset, MediaPackage must have permissions to write to the S3 bucket where the asset will be stored. MediaPackage must also have `GetObject` and `HarvestObject` access to the MediaPackage v2 endpoint where the asset will be harvested from.

You can use the MediaPackage console, the AWS CLI, or the MediaPackage API to create a harvest job. For information, see the [MediaPackage V2 Live API Reference](#).

When you're creating a harvest job, don't put sensitive identifying information like account numbers into free-form fields, such as the ID field. Any data that you enter into MediaPackage might get picked up for inclusion in diagnostic logs or CloudWatch Events.

To create a harvest job

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.
2. Choose **Channel group**.
3. Choose a channel.
4. Choose the **Harvest jobs** table in the channel view.
5. Choose **Create harvest job**.
6. Enter a **Harvest job name**. You can reuse the name after the harvest job expires from your account. The harvest job expires within 15 days of job creation.
7. For **Channel group name**, select the channel group that serves the live stream that you're harvesting the live-to-VOD asset from. By default, you can have up to 10 active harvest jobs within a channel group.
8. For **Channel name**, select the channel that serves the live stream that you're harvesting the live-to-VOD asset from.
9. For **Origin endpoint name**, select the endpoint that serves the live stream that you're harvesting the live-to-VOD asset from. Note the following considerations:

- Your harvest job start time must fall within your MediaPackage endpoint's startover window. The startover window determines the time frame that assets can be harvested from your endpoint. For example, if your endpoint has a startover window of three days, you can harvest your asset anytime within that time frame. To adjust your endpoint's startover window, see [Viewing an origin endpoint in AWS Elemental MediaPackage](#).
 - Your harvested live-to-VOD asset can have a maximum duration of 24 hours.
 - Your endpoint must serve either clear (unencrypted) or encrypted DASH, HLS, or CMAF content.
10. For **Harvested manifest name**, select the endpoint that serves the live stream that you're harvesting the live-to-VOD asset from. You can set up the manifest filter in your filter configuration, but not clip start time, start time, end time, or time delay seconds. For more information, see [FilterConfiguration](#) in the *MediaPackage V2 Live API Reference*.
 11. For **Schedule configuration**, select a **Start date and time** and **End date and time**. The maximum time range is 24 hours.
 - Select a date and time for **When the live-to-VOD asset starts**. The asset's begin time must be at the same time or after the live event started, and it must be in the past. The start time must also be within the startover window on the endpoint. If the endpoint has a window of 5 hours and the start time is 6 hours ago, the harvest job fails.
 - Select a date and time for **When the live-to-VOD asset ends**. The length of the asset can't exceed the startover window on the endpoint. The end time can be in the future.
 12. For **Destination**, select the **S3 bucket name** in which to store the live-to-VOD asset. The bucket must be in the same AWS Region that MediaPackage is harvesting from.
 13. For **S3 destination path**, enter the path to the asset in the bucket. Include the file name for the multivariant playlist of the asset. If the directory structure doesn't already exist in the bucket, MediaPackage creates it.
 14. Choose **Create harvest job**.

Viewing harvest jobs

You can view all harvest jobs that you created within the last 15 days. After that, harvest jobs expire from your account.

To view a harvest job

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.
2. Choose **Channel group**.
3. Choose a channel.
4. Choose the **Harvest jobs** tab.
5. Choose a harvest job name to view the details.

Error messages for failed jobs will be returned in the ErrorMessage field.

Canceling a harvest job

You can cancel a harvest job that is in Queued or In progress status. This stops MediaPackage from moving content into your S3 bucket.

To cancel a harvest job

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.
2. Choose **Channel group**.
3. Choose a channel.
4. Choose the **Harvest jobs** tab.
5. Choose a harvest job that is in Queued or In progress status.
6. Choose **Cancel harvest job**.

AWS Elemental MediaPackage features and functionality

The following sections describe the configurations that are available in AWS Elemental MediaPackage and how they work.

Topics

- [DASH in AWS Elemental MediaPackage](#)
- [CMAF ingest in AWS Elemental MediaPackage](#)
- [A/B forensic watermarking in AWS Elemental MediaPackage](#)
- [Content encryption and DRM in AWS Elemental MediaPackage](#)
- [CMSD headers from AWS Elemental MediaPackage](#)
- [Working with cross-Region failover in AWS Elemental MediaPackage](#)
- [DASH manifest options in AWS Elemental MediaPackage](#)
- [HLS and LL-HLS in AWS Elemental MediaPackage](#)
- [Dynamic Multiview in AWS Elemental MediaPackage](#)
- [Manifest filtering from AWS Elemental MediaPackage](#)
- [Leveraging media quality scores with AWS Elemental MediaPackage](#)
- [Passing through metadata from AWS Elemental MediaPackage](#)
- [MSS in AWS Elemental MediaPackage](#)
- [AWS Elemental MediaPackage rendition groups reference](#)
- [Reset for AWS Elemental MediaPackage channels and endpoints](#)
- [SCTE-35 message options in AWS Elemental MediaPackage](#)
- [Time-shifted viewing with AWS Elemental MediaPackage](#)
- [Enabling trick-play in AWS Elemental MediaPackage](#)

DASH in AWS Elemental MediaPackage

MediaPackage supports Dynamic Adaptive Streaming over HTTP (DASH) for delivering content to a wide range of devices. DASH is an adaptive bitrate streaming protocol that enables high-quality streaming experiences by dynamically adapting to changing network conditions.

DASH features and capabilities

When using DASH in MediaPackage, you can take advantage of the following features:

- **Container type:** DASH requires the CMAF container type.
- **Multi-period support:** MediaPackage supports multi-period DASH manifests, which enable advanced features like ad insertion, content switching, and DRM key rotation.
- **Segment template formats:** MediaPackage supports the Number with timeline segment template format, which uses the `$Number$` variable to refer to segments in the `media` attribute of the `SegmentTemplate` tag.
- **UTC timing options:** MediaPackage provides multiple options for synchronizing players to coordinated universal time (UTC), including UTC Direct, HTTP Head, HTTP Iso, and HTTP Xsdate.
- **SCTE-35 ad markers:** MediaPackage supports signaling ad markers in DASH manifests using either Binary or XML formats.
- **Encryption:** DASH supports both CENC and CBCS encryption methods with PlayReady, Widevine, and Irdeto (CENC only) DRM systems.

DASH manifest structure

A DASH manifest, also known as a Media Presentation Description (MPD), is an XML document that describes the available media segments, their relationships, and other characteristics. The main components of a DASH manifest include:

- **MPD:** The root element containing metadata about the presentation, including profiles, type (static or dynamic), and availability start time.
- **Period:** Represents a period of time in the media presentation. A manifest can contain multiple periods for features like ad insertion.
- **AdaptationSet:** Groups related media content, such as different encodings of the same audio or video content.
- **Representation:** Describes a specific version of the content, such as a particular bitrate or resolution.
- **SegmentTemplate:** Defines how to construct URLs for media segments and provides timing information.

MediaPackage generates DASH manifests that conform to the DASH-IF Interoperability Guidelines, ensuring broad compatibility with DASH players.

DASH period triggers

MediaPackage allows you to configure how periods are created in DASH manifests. You can choose from the following period triggers:

- **Avails:** Creates new periods when SCTE-35 ad markers that pass the configured filter are encountered.
- **DRM key rotation:** Creates new periods when encryption keys are rotated.
- **Source changes:** Creates new periods when there are changes in the stream set.
- **Source disruptions:** Creates new periods when there are gaps in all content streams.
- **None:** Formats the manifest as a single period, only creating additional periods when DRM settings change.

For more information about multi-period DASH, see [Multi-period DASH in AWS Elemental MediaPackage](#).

DASH use cases

Consider using DASH endpoints in the following scenarios:

- Delivering content to Android devices and smart TVs that support DASH
- Implementing advanced advertising workflows with multi-period DASH
- Supporting DRM-protected content with PlayReady, Widevine, or Irdeto
- Creating multi-protocol streaming workflows that need to include DASH alongside other formats

For information about creating DASH endpoints, see [Create a DASH manifest](#).

CMAF ingest in AWS Elemental MediaPackage

AWS Elemental MediaPackage Version 2 (v2) supports CMAF ingest. CMAF ingest delivers content to viewers in multiple streaming formats from a single ingest workflow. CMAF ingest provides flexibility and features to optimize your streaming experience.

The topics in this section cover CMAF ingest.

Choosing a MediaLive output group for MediaPackage CMAF ingest

When configuring AWS Elemental MediaLive to send content to MediaPackage, you can choose between two output group types for CMAF ingest. Your choice affects which capabilities are available for stream control and presentation in your streaming workflow.

When to use each output group

Use this guidance to determine the output group that best fits your streaming requirements:

When to use CMAF Ingest output group

- You have a simpler workflow that doesn't require custom stream ordering or naming
- You want to configure the system simply with minimal setup
- Stream order and custom properties are not critical for your use case

When to use MediaPackage output group

- You need additional passthrough capabilities for stream metadata
- You want to preserve the exact order of streams as configured in MediaLive
- You need enhanced control over stream presentation and properties

How each output group works

CMAF Ingest output group

The CMAF Ingest output group provides straightforward CMAF ingest with minimal configuration. MediaPackage extracts metadata from the initialization segments and uses MP4 segments to generate manifests. This output group requires simpler setup but provides standard stream handling.

MediaPackage output group

The MediaPackage output group enables additional passthrough capabilities and desired stream ordering when customers choose this output group instead of the CMAF Ingest output group. This provides enhanced control over how streams are presented in output manifests.

The following table compares the capabilities available with each MediaLive output group:

Capability comparison between MediaLive output groups

Capability	CMAF Ingest Output Group	MediaPackage Output Group
Stream order preservation	Standard (alphabetical order in output manifest)	Enhanced (preserves desired stream ordering)
Stream metadata passthrough	Standard	Enhanced (additional passthrough capabilities)
Setup complexity	Simple	More configuration options

Enhanced CMAF capabilities with MediaPackage output groups

When using the MediaPackage output group in MediaLive, AWS Elemental MediaPackage provides additional passthrough capabilities and desired stream ordering. These enhanced capabilities give you greater control over how streams are presented to viewers in your streaming applications.

Stream ordering preservation

When you choose the MediaPackage output group in MediaLive, the desired stream ordering configured in MediaLive is preserved in the output manifests. This allows you to control how different video, audio, and subtitle streams appear to viewers.

Stream ordering preservation helps you:

- Ensure viewers start with your intended default streams
- Present quality choices in your preferred order
- Organize streams logically for better viewer navigation

Additional passthrough capabilities

The MediaPackage output group provides additional passthrough capabilities that enable enhanced stream metadata to be preserved from MediaLive to the output manifests.

These capabilities support:

- Enhanced stream identification and presentation
- Improved viewer experience through better stream organization

- More control over how streams appear in player interfaces

Output locking mode for CMAF channels

When you create a CMAF channel in AWS Elemental MediaPackage, you choose an output locking mode. Use this mode to control how segments are aligned and presented in output manifests. The output locking mode affects segment duration, media sequence numbering, and DRM key rotation behavior.

Understanding output locking modes

MediaPackage supports two output locking modes for CMAF channels:

Epoch locked (default)

Epoch-locked mode synchronizes segment boundaries across channels relative to a fixed epoch reference point. Use this mode when you need cross-region synchronization or failover capabilities. This is the default behavior for CMAF channels.

Non-epoch locked

Non-epoch-locked mode decouples segment alignment from the epoch reference point. Use this mode for single-region workflows where you don't need cross-region synchronization and you want predictable segment durations and target duration values in output manifests.

Important

You can't change the output locking mode after you create a channel. To switch modes, you must create a new channel.

When to use each mode

Consider the following factors to choose the output locking mode for your workflow:

When to use epoch-locked mode

- You need cross-region synchronization or failover.
- You have redundant channels in multiple AWS Regions that must produce identical manifests.
- You want DRM key rotation at fixed intervals from a common reference point.

When to use non-epoch-locked mode

- You have a single-region workflow that doesn't require cross-region failover.
- You want the HLS target duration to be the same as the configured output segment duration.
- You don't need to maintain epoch alignment, which can create large segments for the service to maintain alignment with epoch boundaries.
- You want your upstream encoder to be able to change frame rate or input segment duration during a session.
- You want DRM key rotation based on the endpoint's last modified time instead of a fixed epoch.

Behavior differences between modes

The following table compares how each output locking mode affects manifest and segment behavior:

Comparison of output locking modes

Behavior	Epoch locked	Non-epoch locked
HLS EXT-X-TARGETDURATION	Inflates up to 2x the configured segment duration when SCTE is enabled.	Equals the configured segment duration.
HLS MediaSequenceNumber	The service derives this value from the epoch reference.	Starts at 0 and increases monotonically.
Segment durations	Can produce large combined segments for epoch alignment.	Cluster around the configured segment duration value.
DASH period start times	Aligned to epoch reference.	Uses actual start times.
DRM key rotation interval	Fixed intervals from epoch 0.	Fixed intervals from endpoint or DRM configuration last modified time.
Encoder frame rate and input segment duration changes	Not supported during a session.	Supported during a session.

Behavior	Epoch locked	Non-epoch locked
Cross-region synchronization	Supported.	Not supported.

Output timestamp mode for non-epoch-locked channels

For non-epoch-locked CMAF channels, you can configure an output timestamp mode on the origin endpoint's segment configuration. Use this setting to control how presentation timestamp (PTS) values appear in output segments.

MediaPackage supports the following output timestamp modes:

Passthrough (default)

Preserves the raw PTS values from the input stream without modification.

Rebased to channel start

Rebases output PTS values to start close to 0 relative to the first input segment. This produces smaller PTS values in output segments.

Note

The output timestamp mode is only available for non-epoch-locked CMAF channels. You can't change this setting after you create the origin endpoint.

If you use the rebased-to-channel-start mode and the upstream encoder restarts, we recommend that you reset the origin endpoint. For more information about resetting, see [Resetting channel history in AWS Elemental MediaPackage](#).

A/B forensic watermarking in AWS Elemental MediaPackage

AWS Elemental MediaPackage supports A/B forensic watermarking workflows for CMAF ingest. In an A/B watermarking workflow, the upstream encoder produces two watermark variants of the content and, on each incoming segment, signals which watermark variant and bit position the segment carries. MediaPackage uses this signal to make watermark-aware segment-combining decisions so that watermark bit positions stay aligned in the output. This alignment lets a

downstream content delivery network (CDN) derive the correct bit position for each output segment.

To use A/B watermarking with MediaPackage, your workflow must meet the channel requirement and the upstream encoder must send the `WMPaceInfoIngest` header, as described in the following sections.

You configure A/B watermarking on the upstream encoder. When you use AWS Elemental MediaLive as the encoder, MediaLive produces the A and B watermark variants and sends each variant to a separate MediaPackage destination that you pair. For information about enabling A/B watermarking and configuring the A and B destinations in MediaLive, see [Creating A/B forensic video watermarks](#) in the *AWS Elemental MediaLive User Guide*.

Channel requirements

Important

A/B watermarking is supported only on CMAF ingest channels that use *epoch-locked* output locking mode, which is the default mode for CMAF channels. Watermark-aware segment combining depends on epoch-locked segment numbering to keep watermark bit positions aligned. Channels that use non-epoch-locked mode, and ingest types other than CMAF, don't support A/B watermarking. For more information about output locking modes, see [Output locking mode for CMAF channels](#).

For watermark bit positions to remain aligned across combined segments, configure your workflow so that the origin endpoint segment duration is a whole-number multiple of the encoder output segment duration. Set the encoder's watermark poly period so that the encoder output segment duration multiplied by the poly period equals the origin endpoint segment duration. This produces one watermark bit position per output segment. For example, a 2-second encoder output segment duration with a poly period of 3 matches a 6-second origin endpoint segment duration, and MediaPackage combines three input segments into each output segment.

Note

You can use SCTE-35 ad markers on channels that use A/B watermarking. MediaPackage continues to split segments at SCTE-35 markers, and epoch locking keeps the A and B variants aligned across those splits.

How watermarking affects segment combining

When watermarking is active, MediaPackage keeps each output segment within a single watermark bit position. If the watermark variant or bit position changes between input segments, or the encoder marks a watermark aggregation boundary by using the `firstpart` or `lastpart` fields, MediaPackage ends the current output segment at that boundary instead of combining across it. As a result, most output segments match your configured segment duration, but an output segment can be shorter where a watermark boundary falls. This appears in your HLS and DASH manifests as an occasional shorter segment at those boundaries.

WMPaceInfoIngest header

When A/B watermarking is active, the upstream encoder must include a `WMPaceInfoIngest` HTTP header on each incoming media segment. This header carries the watermark pacing information for the segment, following the [ETSI TS 104 002](#) specification. MediaPackage parses this header to determine how to combine segments while preserving watermark bit positions.

The value of the `WMPaceInfoIngest` header is a JSON object. All of the following fields are required, and the header value must not exceed 500 characters.

WMPaceInfoIngest header fields

Field	Type	Description
<code>version</code>	Integer	The version of the watermark pacing information. The only supported value is 1.
<code>variant</code>	Integer	An integer, assigned by the encoder, that identifies which watermark variant (for example, the A or B rendition) the segment belongs to.
<code>position</code>	Integer	The forensic watermark bit position that the segment carries, as determined by the encoder. This is the watermark bit position, not the segment sequence number. Valid values are -1 through 32767, where -1 indicates that the segment isn't watermarked.

Field	Type	Description
firstpart	Boolean	Whether this segment is the first segment of a watermark aggregation group.
lastpart	Boolean	Whether this segment is the last segment of a watermark aggregation group.

The following is an example WMPaceInfoIngest header value:

```
{"version":1,"variant":0,"position":1024,"firstpart":true,"lastpart":false}
```

Content encryption and DRM in AWS Elemental MediaPackage

Protect your content from unauthorized use through content encryption and digital rights management (DRM). AWS Elemental MediaPackage uses the [AWS Secure Packager and Encoder Key Exchange \(SPEKE\) API](#) to facilitate content encryption and decryption by a DRM provider. Using SPEKE, the DRM provider supplies encryption keys to MediaPackage through the SPEKE API. The DRM provider also supplies licenses to supported media players for decryption. For more information about how SPEKE is used with services and features running in the cloud, see [AWS cloud-based architecture](#) in the *Secure Packager and Encoder Key Exchange API Specification guide*.

Limitations and requirements

When implementing content encryption for MediaPackage, refer to the following limitations and requirements:

- Use the AWS Secure Packager and Encoder Key Exchange (SPEKE) API to facilitate integration with a digital rights management (DRM) system provider. For information about SPEKE, see [What is Secure Packager and Encoder Key Exchange?](#)
- Your DRM system provider must support SPEKE. For a list of DRM providers that support SPEKE, see the [Get on board with a DRM platform provider](#) topic in the *AWS Elemental MediaPackage User Guide*. Your DRM provider can help you set up DRM encryption use in MediaPackage.
- Use MediaPackage to encrypt live content.

Container and DRM system support with SPEKE

MediaPackage supports [SPEKE Version 2.0](#) which uses multiple, distinct encryption keys for audio and video tracks and uses [Content Protection Information Exchange \(CPIX\) Version 2.3](#). For more information about SPEKE Version 2.0 encryption configurations, see [Encryption presets in AWS Elemental MediaPackage](#).

Supported containers and DRM systems

The following table lists the different containers and digital rights management (DRM) systems that SPEKE Version 2.0 supports.

SPEKE Version 2.0 – Support matrix for container and DRM system	Apple FairPlay	ClearKey AES-128	Google Widevine	Microsoft PlayReady	Irdeto
TS container	✓ Supports SAMPLE-AES	✓ Supports AES-128	Not supported	Not supported	Not supported
CMAF container	✓ Supports cbcs encryption	Not supported	✓ Supports cbcs and cenc encryption	✓ Supports cbcs and cenc encryption	✓ Supports cenc encryption

Supported DRM system IDs

The following table lists the different DRM [system IDs](#) that MediaPackage supports.

System IDs – Support matrix for DRM system	Apple FairPlay	ClearKey AES-128	Google Widevine	Microsoft PlayReady	Irdeto
	94ce86fb- 07ff-4f43- adb8-93d 2fa968ca2	3ea8778f- 7742-4bf9 -b18b-e83 4b2acbd47	edef8ba9- 79d6-4ace -a3c8-27d cd51d21ed	9a04f079- 9840-4286 -ab92-e65 be0885f95	80a6be7e- 1448-4c37 -9e70-d5a ebe04c8d2

Deploying SPEKE

Your digital rights management (DRM) system provider can help you get set up to use DRM encryption in MediaPackage. Generally, the provider gives you a SPEKE gateway to deploy in your AWS account in the same AWS Region where MediaPackage is running. For information about configuring encryption settings for your endpoint, see [encryption fields](#).

If you must build your own API Gateway to connect MediaPackage to your key service, you can use the [SPEKE Reference Server](#) available on GitHub as a starting point.

The following sections provide guidance on how to implement content encryption using SPEKE for MediaPackage.

Topics

- [Implementing SPEKE v2.0 with AWS Elemental MediaPackage](#)
- [Preparing and managing certificates for content key encryption](#)
- [AWS Elemental MediaPackage key rotation behavior](#)
- [Managing DRM segment metadata in AWS Elemental MediaPackage](#)
- [Excluding DRM session keys in AWS Elemental MediaPackage](#)
- [Encryption presets in AWS Elemental MediaPackage](#)

Implementing SPEKE v2.0 with AWS Elemental MediaPackage

This topic supplements the [SPEKE API v2 — Customizations and constraints to the DASH-IF specification](#) with practical implementation guidance verified through production integration testing with AWS Elemental MediaPackage v2.

Note

This guide does not cover [SPEKE API v2 — Content Key Encryption](#). Examples show content keys returned in `pskc:PlainValue` (cleartext base64), protected by HTTPS transport encryption.

Topics

- [HTTP protocol requirements](#)
- [CPIX document structure](#)
- [DRM system signaling](#)
- [Key delivery to players](#)
- [Testing and verification](#)
- [Implementation checklist](#)

HTTP protocol requirements

Request headers (from MediaPackage)

Header	Required	Value
Content-Type	Yes	application/xml
X-Speke-Version	Yes	2.0

Response headers (from key provider)

Header	Required	Value
Content-Type	Yes	application/xml

Header	Required	Value
X-Speke-Version	Yes	Echo back the request value
X-Speke-User-Agent	Yes	Your server identifier

Error responses (HTTP 422)

Condition	Error message
Missing contentId	Missing CPIX@contentId
Missing version	Missing CPIX@version
Unsupported version	Unsupported CPIX@version
Missing encryption scheme	Missing ContentKey@commonEncryptionScheme for KID {id}
Mixed encryption schemes	Non compliant ContentKey@commonEncryption Scheme combination
Unsupported SPEKE version	Unsupported SPEKE version

CPIX document structure

Every SPEKE v2.0 exchange uses CPIX v2.3 XML with the namespaces `xmlns:cpix="urn:dashif:org:cpix"` and `xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc"`.

Root element: cpix:CPIX

Attribute	Required	Description
contentId	Yes	Content identifier from MediaPackage. Must not be empty.

Attribute	Required	Description
version	Yes	Must be 2.3. Return error if missing or unsupported.

The key provider must NOT modify `contentId` or `version` in the response. However, the key provider may override the `kid` value on `ContentKey` elements — see [Overriding the key identifier](#).

ContentKeyList (mandatory)

Contains one or more `ContentKey` elements. Each represents an encryption key.

Attribute	Required	Description
kid	Yes	Key ID in UUID format
commonEncryptionScheme	Yes	One of: <code>cenc</code> , <code>cbc1</code> , <code>cbcs</code>
explicitIV	Optional	Base64-encoded 16-byte initialization vector

All `ContentKey` elements in a single request must use the same `commonEncryptionScheme`. The response must include `cpix:Data/pskc:Secret/pskc:PlainValue` with a base64-encoded 16-byte AES key.

DRMSystemList (mandatory)

Contains one `DRMSystem` element per key per DRM system. Example: 2 keys x 3 DRM systems = 6 `DRMSystem` elements.

Attribute	Required	Description
kid	Yes	Key ID this DRM system applies to
systemId	Yes	DRM system UUID

Mandatory HLS signaling rules:

1. All `HLSSignalingData` content must be base64-encoded.
2. Both `playlist="media"` and `playlist="master"` must be present.
3. Decoded media and master tags must have identical attributes. Only the tag name differs.
4. HLS tags must NOT contain a `SCHEME` attribute.
5. Each `DRMSystem` must reference exactly one `kid` with exactly one HLS tag.

ContentKeyPeriodList (live streaming)

Must be echoed back unchanged from the request. MediaPackage sends this even without key rotation.

ContentKeyUsageRuleList (mandatory)

Defines which keys protect which tracks. The AWS SPEKE v2 spec explicitly requires the key provider to echo back the `ContentKeyUsageRuleList` unchanged — MediaPackage defines the encryption contract (which keys protect which tracks), and the DRM server must not alter it.

Attribute	Required	Description
<code>kid</code>	Yes	Key ID this rule applies to
<code>intendedTrackType</code>	Yes	ALL, VIDEO, AUDIO, SD, HD, UHD, etc.

At least one `VideoFilter` or `AudioFilter` child element is required.

DRM system signaling

System IDs

System ID	Reference	Encryption Schemes
94ce86fb-07ff-4f43-adb8-93d2fa968ca2	FairPlay	cbcs only
edef8ba9-79d6-4ace-a3c8-27dcd51d21ed	Widevine	cenc, cbcs

System ID	Reference	Encryption Schemes
9a04f079-9840-4286-ab92-e65 be0885f95	PlayReady	cenc, cbcs
3ea8778f-7742-4bf9-b18b-e83 4b2acbd47	Clear Key AES-128	cbc1 only

FairPlay signaling

Response XML structure:

```
<cpix:DRMSystem kid="8fe6f4de-..." systemId="94ce86fb-07ff-4f43-adb8-93d2fa968ca2">
  <cpix:HLSSignalingData playlist="media">BASE64_ENCODED_EXT_X_KEY</
cpix:HLSSignalingData>
  <cpix:HLSSignalingData playlist="master">BASE64_ENCODED_EXT_X_SESSION_KEY</
cpix:HLSSignalingData>
</cpix:DRMSystem>
```

FairPlay uses only HLS signaling (no PSSH or ContentProtectionData).

Decoded HLS signaling (media and master identical except tag name):

```
#EXT-X-KEY:METHOD=SAMPLE-AES,URI="skd://"
<BASE64_JSON>","KEYFORMAT="com.apple.streamingkeydelivery",KEYFORMATVERSIONS="1"
#EXT-X-SESSION-KEY:METHOD=SAMPLE-AES,URI="skd://"
<BASE64_JSON>","KEYFORMAT="com.apple.streamingkeydelivery",KEYFORMATVERSIONS="1"
```

The skd:// URI contains a base64-encoded JSON payload — NOT just the kid hex:

```
{"ContentId":"101","KeyId":"8fe6f4de-841a-3acd-a81a-be516f356bbb","IV":"base64-encoded-IV=="}
```

Important

The KeyId must use UUID format with hyphens. Do NOT include a separate IV=0x... attribute in the HLS tag — the IV is inside the JSON payload only.

Widevine signaling

Response XML structure:

```
<cpix:DRMSystem kid="dda7afc9-..." systemId="edef8ba9-79d6-4ace-a3c8-27dcd51d21ed">
  <cpix:PSSH>BASE64_ENCODED_PSSH_BOX</cpix:PSSH>
  <cpix:ContentProtectionData>BASE64_ENCODED_CENC_PSSH_XML</cpix:ContentProtectionData>
  <cpix:HLSSignalingData playlist="media">BASE64_ENCODED_EXT_X_KEY</
cpix:HLSSignalingData>
  <cpix:HLSSignalingData playlist="master">BASE64_ENCODED_EXT_X_SESSION_KEY</
cpix:HLSSignalingData>
</cpix:DRMSystem>
```

Widevine includes PSSH box, ContentProtectionData, and HLS signaling. All values are base64-encoded.

PSSH Box: Base64-encoded binary data containing the Widevine PSSH box. The box structure is:

```
[4 bytes] Box size (big-endian uint32)
[4 bytes] Box type: "pssh"
[1 byte] Version: 0
[3 bytes] Flags: 0
[16 bytes] System ID: edef8ba9-79d6-4ace-a3c8-27dcd51d21ed (Widevine)
[4 bytes] Data size (big-endian uint32)
[N bytes] Widevine PSSH data (protobuf-encoded, contains key IDs and provider)
```

ContentProtectionData (base64-decoded): Must use the standard CENC PSSH XML format:

```
<cenc:pssh xmlns:cenc="urn:mpeg:cenc:2013">PSSH_BOX_BASE64</cenc:pssh>
```

The PSSH_BOX_BASE64 inside cenc:pssh is the same base64-encoded PSSH box as in the cpix:PSSH element.

Decoded HLS signaling: The URI must be a data URI with the PSSH box embedded inline:

```
#EXT-X-KEY:METHOD=SAMPLE-AES,URI="data:text/
plain;base64,<PSSH_BOX_BASE64>","KEYFORMAT="urn:uuid:edef8ba9-79d6-4ace-
a3c8-27dcd51d21ed",KEYFORMATVERSIONS="1"
```

⚠ Important

The URI must be a `data:text/plain;base64`, data URI containing the PSSH box — NOT a license server URL (e.g., `https://widevine.example.com/license`). ContentProtectionData must use the `cenc:pssh` format — NOT custom wrappers like `<ContentProtectionData><PSSH>...</PSSH></ContentProtectionData>`. Do NOT include a SCHEME attribute in HLS tags.

PlayReady signaling**Response XML structure:**

```
<cpix:DRMSystem kid="566209fc-..." systemId="9a04f079-9840-4286-ab92-e65be0885f95">
  <cpix:PSSH>BASE64_ENCODED_PSSH_BOX</cpix:PSSH>
  <cpix:ContentProtectionData>BASE64_ENCODED_CENC_PSSH_AND_MSPR_PRO</
cpix:ContentProtectionData>
  <cpix:HLSSignalingData playlist="media">BASE64_ENCODED_EXT_X_KEY</
cpix:HLSSignalingData>
  <cpix:HLSSignalingData playlist="master">BASE64_ENCODED_EXT_X_SESSION_KEY</
cpix:HLSSignalingData>
</cpix:DRMSystem>
```

PlayReady includes PSSH box, ContentProtectionData (with both CENC and PRO elements), and HLS signaling.

PSSH Box: Base64-encoded binary data containing the PlayReady PSSH box. The box structure is:

```
[4 bytes] Box size (big-endian uint32)
[4 bytes] Box type: "pssh"
[1 byte] Version: 0
[3 bytes] Flags: 0
[16 bytes] System ID: 9a04f079-9840-4286-ab92-e65be0885f95 (PlayReady)
[4 bytes] Data size (big-endian uint32)
PlayReady Object (PRO) record:
  [4 bytes] Record count (little-endian uint32, always 1)
  [4 bytes] Record size (little-endian uint32)
  [N bytes] WRM Header XML (UTF-16LE encoded)
```

The WRM Header XML contains KID, CHECKSUM, LA_URL (license acquisition URL), ALGID, and KEYLEN elements for each content key.

ContentProtectionData (base64-decoded): Must contain BOTH `cenc:pssh` AND `mspr:pro` elements:

```
<cenc:pssh xmlns:cenc="urn:mpeg:cenc:2013">PSSH_BASE64</cenc:pssh>
<mspr:pro xmlns:mspr="urn:microsoft:playready">PRO_RECORD_BASE64</mspr:pro>
```

The `cenc:pssh` contains the full PSSH box. The `mspr:pro` contains the PlayReady Object record (count + size + WRM Header XML in UTF-16LE).

Decoded HLS signaling (attribute order: METHOD, KEYFORMAT, KEYFORMATVERSIONS, URI):

The data URI must use `charset=UTF-16` and contain the PlayReady Object record (PRO), NOT the raw PSSH box:

```
#EXT-X-KEY:METHOD=SAMPLE-
AES,KEYFORMAT="com.microsoft.playready",KEYFORMATVERSIONS="1",URI="data:text/
plain;charset=UTF-16;base64,<PRO_RECORD_BASE64>"
```

Important

The data URI must use `charset=UTF-16` and embed the PlayReady Object record, NOT the raw PSSH box. The PRO record structure: count(4 bytes LE) + size(4 bytes LE) + WRM Header XML (UTF-16LE). Attribute order must be METHOD, KEYFORMAT, KEYFORMATVERSIONS, URI. ContentProtectionData must include BOTH `cenc:pssh` AND `mspr:pro`. Omitting either will cause failures. Do NOT include a SCHEME attribute.

Clear Key AES-128 signaling

Response XML structure:

```
<cpix:DRMSystem kid="136436ea-..." systemId="3ea8778f-7742-4bf9-b18b-e834b2acbd47">
  <cpix:HLSSignalingData playlist="media">BASE64_ENCODED_EXT_X_KEY</
cpix:HLSSignalingData>
  <cpix:HLSSignalingData playlist="master">BASE64_ENCODED_EXT_X_SESSION_KEY</
cpix:HLSSignalingData>
</cpix:DRMSystem>
```

Clear Key AES-128 uses only HLS signaling (no PSSH or ContentProtectionData).

Decoded HLS signaling:

```
#EXT-X-KEY:METHOD=AES-128,URI="https://<key_server>/client/<content_id>/<kid>",KEYFORMAT="identity",KEYFORMATVERSIONS="1"
```

The key delivery endpoint must return the raw 16-byte binary key (not base64) with CORS headers. No DRM license server is needed — the player fetches the key directly.

Key delivery to players

DRM System	Player-side key delivery	License server required
Clear Key AES-128	Player GETs raw 16-byte key from HTTPS URL in manifest	No
FairPlay	Player contacts FPS license server using <code>skd://</code> URI	Yes (Apple FPS license)
Widevine	Player uses EME API with PSSH from data URI	Yes (Google/3rd party)
PlayReady	Player uses EME API with PRO from data URI	Yes (Microsoft/3rd party)

Testing and verification

Verify your SPEKE v2.0 implementation

1. **Test SPEKE endpoint directly.** Send a POST request with a CPIX XML body and verify the response contains `contentId`, `version="2.3"`, `PlainValue` with 16-byte base64 key, `explicitIV`, base64-encoded `HLSSignalingData`, and echoed `ContentKeyPeriodList`.
2. **Verify key determinism.** Send the same request twice. The `PlainValue` should be identical both times.
3. **Verify HLS signaling.** Decode the base64 `HLSSignalingData` and verify: FairPlay uses `skd://` with JSON payload, Widevine uses `data:text/plain;base64,` with PSSH, PlayReady uses `data:text/plain;charset=UTF-16;base64,` with PRO, no `SCHEME` attribute, media and master tags match.

4. **Test with MediaPackage.** Configure a MediaPackage endpoint with SPEKE encryption, start a live stream, check server logs for requests, and verify playback.

Implementation checklist

- HTTP endpoint accepts POST with Content-Type: application/xml
- Validates and echoes X-Speke-Version header
- Returns X-Speke-User-Agent response header
- Validates contentId, version, commonEncryptionScheme
- Returns HTTP 422 with exact error messages per SPEKE specification
- Generates deterministic 16-byte AES keys with explicitIV
- Generates per-kid DRMSystem elements
- Base64-encodes all HLSSignalingData, PSSH, and ContentProtectionData
- FairPlay: skd:// URI contains base64-encoded JSON with ContentId, KeyId (with hyphens), and IV
- Widevine: HLS URI uses data:text/plain;base64,<PSSH>; ContentProtectionData uses cenc:pssh
- PlayReady: HLS URI uses data:text/plain;charset=UTF-16;base64,<PRO>; ContentProtectionData includes both cenc:pssh and mspr:pro; attribute order METHOD, KEYFORMAT, KEYFORMATVERSIONS, URI
- Clear Key AES-128: provides GET endpoint returning raw 16-byte key with CORS headers
- HLS media and master tags have identical attributes; no SCHEME attribute
- Echoes ContentKeyPeriodList and ContentKeyUsageRuleList unchanged
- Includes VideoFilter and/or AudioFilter in usage rules

Preparing and managing certificates for content key encryption

When implementing DRM encryption, AWS Elemental MediaPackage can enhance security by encrypting the content keys within the Content Protection Information Exchange (CPIX) document exchanged with your key provider. This additional encryption layer protects the content keys during transmission between MediaPackage and your DRM key provider.

Overview

When you create or modify origin endpoint resources in MediaPackage for DRM-protected content, you can optionally specify a certificate ARN in the SPEKE encryption settings. This certificate ARN must point to a certificate stored in AWS Certificate Manager (ACM). This certificate provides encryption for content keys delivery between MediaPackage and your key provider, as described in the [SPEKE Version 2.0 content key encryption](#) specification.

This is an opt-in configuration. When you don't provide a certificate in your origin endpoint resource's SPEKE encryption configuration, DRM key providers return content keys as plain text, which is the current and default behavior. When you provide a certificate, the content key returned by your DRM key provider is encrypted using the certificate's public key, and MediaPackage decrypts the encrypted content key with the certificate's private key.

To use encrypted content keys, the following requirements must be met:

- Your DRM key provider must support content key encryption. If you enable this feature for a key provider that doesn't handle content key encryption, playback fails.
- You must provide a certificate stored in AWS Certificate Manager (ACM) in the same Region and Account that you run MediaPackage. For information about ACM, see the [AWS Certificate Manager User Guide](#).

When you create or update an origin endpoint resource with certificate ARN, you must have ACM permissions.

The certificate must meet the following requirements:

- **Status:** The certificate must be in **ISSUED** status
- **Domain name:** The certificate must have a domain name
- **Key algorithm:** The certificate must use RSA-2048 key algorithm
- **Signature algorithm:** The certificate must use either SHA-256 or SHA-512 signature algorithm
- **Region:** The certificate must be in the same AWS Region as your origin endpoint
- **Account:** The certificate must be in the same AWS Account as your origin endpoint
- The origin endpoint ARN length must not exceed 222 characters. An origin endpoint ARN follows this format:
`arn:aws:mediapackagev2:region:123456789012:channelGroup/channelGroupName/channel/channelName/originEndpoint/originEndpointName`. If the ARN length exceeds

this limit, origin endpoint creation with content key encryption will fail due to an ARN length restriction.

The following procedures describe how to prepare and manage the certificate.

To prepare a certificate for DRM content key encryption

1. Open the ACM console at <https://console.aws.amazon.com/acm/>.
2. Import or request a certificate into ACM according to the instructions at [AWS Certificate Manager User Guide](#). Note the resulting certificate ARN because you will need it later.

For use in DRM encryption, your certificate must have a status of **Issued** in ACM.

To use a certificate in MediaPackage

When you create or modify an origin endpoint with DRM encryption, provide your certificate ARN in the SPEKE encryption parameters. This enables content key encryption. You can use the same certificate ARN for multiple origin endpoints. For information, see the encryption settings information in [the section called "Working with endpoints"](#).

How content key encryption works

When you configure certificate-based content key encryption, the following process occurs:

1. **MediaPackage retrieves your certificate:** MediaPackage uses your certificate ARN to retrieve the certificate details from ACM.
2. **CPIX request with certificate:** MediaPackage includes your certificate's public key in the CPIX request sent to your DRM key provider.
3. **DRM provider encrypts keys:** Your DRM key provider uses the certificate's public key for content key encryption
4. **MediaPackage decrypts keys:** MediaPackage uses the certificate's private key for content key decryption
5. **Content encryption:** MediaPackage uses the decrypted content keys to encrypt your content

Note

This process follows the [SPEKE Version 2.0 content key encryption](#) specification for secure key exchange. Content key encryption adds an additional layer of security to the delivery of content key without affecting the final content encryption.

To delete a certificate

To delete a certificate from ACM, it must not be associated with any other resources. Delete the certificate ARN from origin endpoint configurations where you have used it, then delete it from ACM.

Note

If you delete a certificate ARN from an active origin endpoint, the endpoint keeps running, but stops using content key encryption.

AWS Elemental MediaPackage key rotation behavior

When you enable key rotation on live content from TS and CMAF origin endpoints, AWS Elemental MediaPackage retrieves content keys before the live content begins. As the content progresses, MediaPackage retrieves new keys at the interval that you set on the origin endpoint, as described in [Encryption fields](#).

If MediaPackage is unable to retrieve the content key, it takes the following actions:

- If MediaPackage successfully retrieved a content key for this endpoint before, it uses the last key that it fetched. This ensures that endpoints that worked previously continue to work.
- If MediaPackage has *not* successfully retrieved a content key for this endpoint before, MediaPackage responds to the playback request with error 404.

Managing DRM segment metadata in AWS Elemental MediaPackage

AWS Elemental MediaPackage includes DRM segment metadata boxes (SEIG and SGPD) in CMAF container segments by default. These boxes contain key rotation metadata that helps players

manage DRM key changes during playback. However, some devices and players don't support these metadata boxes, which can cause compatibility issues.

Overview of SEIG and SGPD boxes

DRM segment metadata boxes serve specific functions in encrypted content delivery:

- **SEIG (Sample Encryption Information Group) boxes** - Contain sample-level encryption information including initialization vectors and subsample encryption patterns.
- **SGPD (Sample Group Description) boxes** - Define the structure and parameters for sample groups, including encryption-related groupings.

These boxes work together to provide detailed encryption metadata at the segment level, enabling fine-grained control over DRM key rotation and sample encryption patterns.

Device compatibility challenges

While SEIG and SGPD boxes provide valuable functionality, they can cause compatibility issues with certain devices and players:

- **MatchPoint and FairPlay devices** - Some devices that use MatchPoint or FairPlay DRM systems may not properly parse or handle these metadata boxes, leading to playback failures.
- **MP browser implementations** - Certain browser-based media players may encounter parsing errors when processing segments with these metadata boxes.
- **Vuze devices** - Some Vuze-based players may not support the additional metadata complexity, resulting in compatibility issues.
- **Legacy FairPlay implementations** - Older FairPlay implementations may not handle the metadata boxes correctly, particularly in combination with other DRM features.

When these compatibility issues occur, players may fail to start playback, encounter buffering problems, or display error messages related to DRM or segment parsing.

When to exclude segment metadata

Consider excluding DRM segment metadata in the following scenarios:

- **Device compatibility requirements** - When your content must play on devices known to have issues with SEIG and SGPD boxes.

- **Simplified DRM workflows** - When you can handle key rotation through media playlist signaling instead of segment-level metadata.
- **Legacy player support** - When supporting older players that don't implement full support for these metadata boxes.
- **Troubleshooting playback issues** - As a diagnostic step when investigating DRM-related playback problems.

Note

Excluding segment metadata doesn't affect other DRM functionality. PSSH (Protection System Specific Header) and TENC (Track Encryption) boxes remain unaffected and continue to provide essential DRM information.

Alternative key rotation methods

When you exclude segment metadata, key rotation can still be handled through alternative methods:

- **Media playlist signaling** - Key rotation information can be communicated through HLS media playlists using EXT-X-KEY tags or DASH manifests using ContentProtection elements.
- **Manifest-level metadata** - DRM key information can be provided at the manifest level rather than within individual segments.
- **Player-side key management** - Players can manage key rotation based on timing information and DRM license responses rather than segment metadata.

These alternative methods ensure that DRM functionality remains intact while improving compatibility with devices that don't support segment-level metadata boxes.

Configuration requirements

The segment metadata exclusion feature has specific requirements and limitations:

- **Container format requirement** - This setting only affects CMAF container formats. TS (Transport Stream) containers don't include SEIG and SGPD boxes, so this setting doesn't apply.
- **Encryption requirement** - The setting is only available when DRM encryption is enabled on the origin endpoint.

- **Default behavior** - By default, MediaPackage includes segment metadata boxes. You must explicitly enable the exclusion setting to omit them.
- **Existing endpoints** - Existing origin endpoints that don't have this setting configured will continue to include segment metadata boxes (default behavior).

To configure this setting using the MediaPackage console, see [Encryption fields](#).

Excluding DRM session keys in AWS Elemental MediaPackage

MediaPackage supports excluding session keys from HLS and LL-HLS multivariant playlists to improve compatibility with legacy clients and provide more granular access control.

Overview

By default, MediaPackage includes EXT-X-SESSION-KEY tags in HLS multivariant playlists when DRM is enabled. These tags allow clients to pre-fetch encryption keys, which can improve playback performance. However, some scenarios require excluding these session keys:

- **Legacy client compatibility** - Some older HLS clients have issues processing session keys and may fail to play content when these tags are present.
- **Access control** - When using manifest filtering to control access to specific content variants, you may want to provide key information only for the streams a client actually has access to, rather than exposing all keys in the session key tags.

When session keys are excluded, encryption key information is still available in the individual media playlists through EXT-X-KEY tags, ensuring that DRM functionality remains intact.

Configuration options

You can exclude session keys using two methods:

- **Static configuration** - Configure the setting on the origin endpoint HLS or LL-HLS manifests to exclude session keys. This is done through the **DRM settings** in the filter configuration. For console instructions, see [Manifest fields](#).
- **Dynamic exclusion** - Use the `aws.drmsettings=exclude_session_keys` query parameter to exclude session keys on a per-request basis. For more information, see [Manifest filtering](#).

Note

If session key exclusion is enabled in the static configuration, it cannot be overridden using query parameters. This follows the standard MediaPackage pattern where static filters take precedence over dynamic parameters.

Encryption presets in AWS Elemental MediaPackage

SPEKE Version 2.0 supports the use of multiple, distinct encryption keys for audio and video tracks. MediaPackage uses **presets** to configure the encryption. The MediaPackage API defines these presets, and they appear in the MediaPackage console in the **Video encryption preset** and **Audio encryption preset** menus of the **Package Encryption endpoints configuration** section. The presets map encryption keys to specific audio or video tracks, based on the number of channels for audio tracks, and based on the video resolution for video tracks. MediaPackage uses specific combinations of audio and video encryption presets to support three different encryption scenarios:

- [Scenario 1: Unencrypted tracks and encrypted tracks](#)
- [Scenario 2: Single encryption key for all audio and video tracks](#)
- [Scenario 3: Multiple encryption keys for audio and video tracks](#)

Scenario 1: Unencrypted tracks and encrypted tracks

You can choose *not* to encrypt the audio or the video tracks by selecting the **UNENCRYPTED** preset in the **Video encryption preset** or the **Audio encryption preset** menus. You can't select **UNENCRYPTED** for both audio and video presets, because doing so would mean that you don't intend to encrypt any of the tracks at all. Also, you can't combine **UNENCRYPTED** and **SHARED** presets for audio and video, because **SHARED** is a special preset. For more information, see [Scenario 2: Single encryption key for all audio and video tracks](#).

The following list describes valid combinations of **UNENCRYPTED** presets:

- **UNENCRYPTED** for audio tracks, and any video preset with a name that starts with **PRESET-VIDEO-**
- **UNENCRYPTED** for video tracks, and any audio preset with a name that starts with **PRESET-AUDIO-**

Scenario 2: Single encryption key for all audio and video tracks

The SPEKE Version 2.0 **SHARED** preset uses a single encryption key for all audio and video tracks, as in SPEKE Version 1.0. When you select the **SHARED** preset, select it for both audio and video encryption.

Scenario 3: Multiple encryption keys for audio and video tracks

When you use a preset with a name that starts with PRESET-VIDEO- or PRESET-AUDIO-, MediaPackage encrypts the audio tracks and video tracks with the number of encryption keys that the specific preset defines. The following tables show how many keys MediaPackage requests from the key server and how those keys map to tracks. If no track matches the criteria for a particular key, MediaPackage does not use that key to encrypt any track.

MediaPackage encrypts I-frame only trickplay tracks with the key corresponding to their resolution.

In the following table, the **Key name** value is the value of the ContentKeyUsageRule@IntendedTrackType attribute that MediaPackage uses in the CPIX document. This is sent to the SPEKE server for a specific content key.

Video encryption presets

Preset name	Number of keys	Key name	Minimum resolution	Maximum resolution
PRESET-VIDEO-1	1	VIDEO	No minimum or maximum resolution. MediaPackage encrypts all tracks with the same key.	
PRESET-VIDEO-2	2	SD	No minimum	<= 1024x576
		HD	> 1024x576	No maximum
PRESET-VIDEO-3	3	SD	No minimum	<= 1024x576
		HD	> 1024x576	<= 1920x1080
		UHD	> 1920x1080	No maximum
PRESET-VIDEO-4	4	SD	No minimum	<= 1024x576

Preset name	Number of keys	Key name	Minimum resolution	Maximum resolution
PRESET-VIDEO-5	5	HD	> 1024x576	<= 1920x1080
		UHD1	> 1920x1080	<= 4096x2160
		UHD2	> 4096x2160	No maximum
		SD	No minimum	<= 1024x576
		HD1	> 1024x576	<= 1280x720
		HD2	> 1280x720	<= 1920x1080
		UHD1	> 1920x1080	<= 4096x2160
		UHD2	> 4096x2160	No maximum
PRESET-VIDEO-6	4	SD	No minimum	<= 1024x576
		HD1	> 1024x576	<= 1280x720
		HD2	> 1280x720	<= 1920x1080
		UHD	> 1920x1080	No maximum
PRESET-VIDEO-7	3	SD+HD1	No minimum	<= 1280x720
		HD2	> 1280x720	<= 1920x1080
		UHD	> 1920x1080	No maximum
PRESET-VIDEO-8	4	SD+HD1	No minimum	<= 1280x720
		HD2	> 1280x720	<= 1920x1080
		UHD1	> 1920x1080	<= 4096x2160
		UHD2	> 4096x2160	No maximum

Preset name	Number of keys	Key name	Minimum resolution	Maximum resolution
SHARED	1	ALL	No minimum or maximum resolution. MediaPackage encrypts all video and audio tracks with the same key.	
UNENCRYPTED	0	N/A	MediaPackage does not encrypt any video track.	

In the following table, the **Key name** value is the value of the ContentKeyUsageRule@IntendedTrackType attribute that MediaPackage uses in the CPIX document. This is sent to the SPEKE server for a specific content key.

Audio encryption presets

Preset name	Number of keys	Key name	Minimum number of channels	Maximum number of channels
PRESET-AUDIO-1	1	AUDIO	No minimum or maximum number of channels. MediaPackage encrypts all audio and video tracks with the same key.	
PRESET-AUDIO-2	2	STEREO_AUDIO	No minimum	2
		MULTICHAN NEL_AUDIO	> 2	No maximum
PRESET-AUDIO-3	3	STEREO_AUDIO	No minimum	2
		MULTICHAN NEL_AUDIO_3_6	> 2	<= 6
		MULTICHAN NEL_AUDIO_7	> 6	No maximum

Preset name	Number of keys	Key name	Minimum number of channels	Maximum number of channels
SHARED	1	ALL	No minimum or maximum number of channels. MediaPackage encrypts all audio and video tracks with the same key.	
UNENCRYPTED	0	N/A	MediaPackage does not encrypt any audio track.	

Now you know how MediaPackage supports SPEKE Version 2.0 presets for unencrypted tracks and encrypted tracks. With these presets, you can use a single encryption key for all audio and video tracks, and multiple encryption keys for audio and video tracks.

CMSD headers from AWS Elemental MediaPackage

AWS Elemental MediaPackage uses Common Media Server Data (CMSD) HTTP response headers to provide downstream systems with information about content coming from an endpoint. These headers are defined by the Consumer Technology Association's CTA-5006 specification that establishes a uniform method for media servers and services to communicate data about media objects.

Media servers such as AWS Elemental MediaPackage use CMSD headers in responses to content requests. The headers can include data about the content, such as the type of content, playback duration, and quality scores. Servers receiving these headers can use the information for things like input switching and logging analysis.

The following sections describe keys that MediaPackage uses in CMSD headers.

Common CMSD keys

The following table describes common CMSD keys that MediaPackage uses. For a full list of keys, see the Consumer Technology Association's [Common Media Server Data CTA-5006 specification](#).

Key name	Value	CMSD header example
n	The identifier for the MediaPackage origin endpoint of this object.	CMSD-Static: n="MediaPackage:<region>:<channelGroup>:<channel>:<endpoint>"
st	The type of content being returned. Supported values are: - l = live content - v = VOD content	CMSD-Static: st=l
sf	The streaming format of the response. Supported values are: - d h = MP4 - h = TS	CMSD-Static: sf=(d h)
ot	The media role of the object being returned. Supported values are: - m = text file (manifest or playlist) - a = audio only - v = video only - av = muxed audio and video - i = init segment - c = caption or subtitle	CMSD-Static: ot=v

Key name	Value	CMSD header example
	• <code>tt</code> = ISOBMFF timed text track • <code>k</code> = cryptographic key, license, or certificate • <code>o</code> = other	
<code>br</code>	The average encoded bitrate of the object.	CMSD-Static: <code>br=1000000</code>
<code>ht</code>	The amount of time that MediaPackage waited to send this response to CloudFront. Measured in milliseconds.	CMSD-Static: <code>ht=1000</code>
<code>d</code>	The playback duration of the object. Measured in milliseconds.	CMSD-Static: <code>d=2000</code>
<code>at</code>	The wallclock time that the first byte of this object became available to the server. Measured in integer milliseconds since the Unix Epoch.	CMSD-Static: <code>at=1656703938470</code>

AWS custom CMSD keys

The following table describes custom AWS keys that MediaPackage uses.

AWS custom key name	Value	CMSD header example
com.amazonaws-mqcs	The media quality confidence score (MQCS) rating for the segment.	CMSD-Static: com.amazonaws-mqcs="90"
com.amazonaws-mqcs-seq	The aggregate of MQCS ratings for all segments in the sequence. Value is from 1-100.	CMSD-Static: com.amazonaws-mqcs-seq="100"
com.amazonaws-mqcs-seq-tracks	The aggregate of MQCS ratings for all segments in the sequence, by object type. Value is from 1-100.	CMSD-Static: com.amazonaws-mqcs-seq-tracks="v:100;a:90;s:50;m:60"

Working with cross-Region failover in AWS Elemental MediaPackage

MediaPackage supports workflows where a CDN transparently fails over between multiple MediaPackage Live v2 origins located in different AWS Regions. The failover functionality also works in a single Region, but such a configuration doesn't protect against regional failures. The purpose of such failover architectures is to force CDN requests to be forwarded to one or more backup origins if the stream on the primary origin is not considered healthy. To be transparent, the failover needs all the MediaPackage origins to produce symmetric streams. This requirement can

be satisfied through the combination of multiple requirements and restrictions at different levels in the workflow as described in the following sections.

Requirements for cross-Region failover in AWS Elemental MediaPackage

If your workflow respects these requirements and restrictions, origin failovers should be seamless for your viewers and not generate live-stream-playback disruptions.

Encoding

The live encoders need to send aligned ingest streams to all involved MediaPackage channels. This is achieved through the use of epoch-locked CMAF ingest streamsets, where segmentation is using a regular cadence, and the indexing of the segments is based on the epoch time. This can be configured in MediaLive and AWS Elemental Live using the CMAF Ingest output group. For more details about the configuration and the input timecode requirements, see [Creating CMAF Ingest Output Groups](#) in the *MediaLive user guide*.

The encoding settings and CMAF Ingest output group settings must be identical across multiple encoders. However, it's possible to use heterogeneous MediaLive channel types in different Regions, such as a standard MediaLive channel in Region A and a single pipeline MediaLive channel in Region B.

Your encoder configuration must also adhere to [Encoder restrictions for cross-Region failover](#).

Packaging origination

All involved MediaPackage Live v2 channels and endpoints need to be configured with strictly identical configuration settings, such as URL path, segmentation parameters, and encryption parameters. The channel input type needs to be CMAF. If the channel input type is HLS, the alignment of the output streams will not be guaranteed. Finally, the MediaPackage channel that will act as the primary origin in the CDN origin group needs to include a **Force endpoint error configuration** that will have MediaPackage return 404 responses to the CDN in some problematic situations, like missing or incomplete ingest, failed key rotations, or slate input (resulting from the upstream encoder having lost its contribution source). For more information, see [Endpoint settings fields](#). We recommend to set the primary MediaPackage origin settings with aggressive **Force endpoint error configuration** settings (at least Stale manifests, incomplete manifest and Slate input). These settings should not be activated on the backup MediaPackage origin, in order to maximize chances of a successful origin failover at the CDN level.

CDN

The CDN distribution should be configured to trigger origin failovers on network connection errors and 404 Not Found response codes returned by the origin. Other 4xx/5xx response codes can also be used as failover triggers, on top of the 404 code, to cover a wider range of problems that are not related to the MediaPackage endpoint error configuration. For more information, see [Optimize high availability with CloudFront origin failover](#).

Encoder restrictions for cross-Region failover

Your encoder configuration, or re-configuration, must adhere to the following restrictions for cross-Region failover to work effectively.

- All video framerates in the CMAF output group need to be consistent. They should be either all fractional framerates, or all integer framerates, not a mix of the two. The combined use of framerates that are multiples of one another, like 25fps and 50fps or 29.97fps and 59.94fps, is allowed.
- The transition from fractional framerates to integer framerates (or the other way around) across two encoding sessions is not allowed. The two framerates also need to be multiples of one another: 25fps to 50fps or 50fps to 25fps are allowed transitions, but 25fps to 30fps or 30fps to 25fps are not.
- The output segments sequence numbers should not go back in the past across two encoding sessions. The segment duration should not increase and the reference Epoch time should not change after the first encoding session.

If segment numbers go backward, MediaPackage sends a 409 error response to the encoder. To recover from this state, see [Reset for channels and endpoints](#).

- The encoder output names must be identical across all Regions.

DASH manifest options in AWS Elemental MediaPackage

This section describes the options that AWS Elemental MediaPackage offers for modifying live output DASH manifests.

Default DASH manifest

The following is a truncated example of a DASH manifest with no treatments:

```

<MPD>
  <Period>
    <AdaptationSet>
      <SegmentTemplate>
        <SegmentTimeline>
          <S />
        </SegmentTimeline>
      </SegmentTemplate>
    <Representation>
    </Representation>
  </AdaptationSet>
  .
  .
</Period>
</MPD>

```

The elements of the DASH manifest are nested within the MPD (media presentation description) object. These are the elements of the manifest:

- **Period** - The entire manifest is nested in one period.
- **AdaptationSet** - Groups together representations of the same type (video, audio, or captions). There are one or more AdaptationSets in the Period.
- **SegmentTemplate** - Defines properties of the AdaptationSet, such as the timescale and access URLs for media and initialization segments.
- **SegmentTimeline** - Describes when each segment is available for playback. There is one SegmentTimeline for each SegmentTemplate.
- **S** - Describes when the segment is available (*t* value), the duration of the segment (*d* value), and a count of how many additional consecutive segments have this same duration (*r* value). There are one or more segments in the SegmentTimeline.
- **Representation** - Describes an audio, video, or captions track. There are one or more Representations in each AdaptationSet. Each representation is a track.

MediaPackage can modify how some of these elements are presented in the output manifest. You can use the following treatment options on the output live manifest:

- Separate the manifest into multiple periods, to permit ad breaks. See [Multi-period DASH in AWS Elemental MediaPackage](#).

- Reduce the length of the manifest to make processing and playback more efficient. See [DASH manifest compactness](#).
- Compress repeating audio segment durations using Segment Duration Patternization (SDP). See [DASH audio timeline pattern](#).

Multi-period DASH in AWS Elemental MediaPackage

The ability to insert multiple periods in DASH manifests for live is available in AWS Elemental MediaPackage.

A period is a chunk of content in the DASH manifest, defined by a start time and duration. MediaPackage can partition the DASH manifest into multiple periods to indicate boundaries between changes in the content. MediaPackage can signal new periods based on several triggers including:

- DRM configuration changes
- Avails
- DRM key rotation
- Source stream changes
- Source disruptions

MediaPackage will always create a period at the beginning of content and whenever DRM configuration changes. The other triggers can be configured per DASH manifest, and have the following effects:

- **Avails** - MediaPackage will create a new period whenever it detects a SCTE-35 message which matches the Segment SCTE Filter configuration as an ad. All SCTE-35 messages will be signaled in the DASH manifest, whether it triggers a new period or not.
- **DRM key rotation** - MediaPackage will create a new period whenever the underlying DRM key rotates. This setting allows MediaPackage to signal the `default_KID` and `pssh` values in the manifest when key rotation is enabled. If this period trigger is not enabled, endpoints with DRM key rotation will not show the `default_KID` and `pssh` values in the manifest.
- **Source stream changes** - MediaPackage will create a new period when it detects stream set changes, allowing all streams to be signaled in the manifest, rather than just those available at manifest publish time.

- **Source disruptions** - MediaPackage will create a new period after source input has been lost and restored, enabling the gap in content to be signaled to the player.

DASH manifest compactness

For some players, processing a manifest with duplicate data about each representation (track) is difficult and slow. To reduce some of the burden, AWS Elemental MediaPackage compacts the manifest by default. Compacting is achieved by moving some attributes from the Representation object to the AdaptationSet object. This way, rather than having the attributes defined for each representation in the manifest, they're defined once at a higher level. The representations then inherit these attributes from the adaptation set.

If you require an uncompacted manifest, you can set the [manifest compactness](#) on the DASH endpoint.

Example standard DVB-DASH manifest (compacted)

In this example, the SegmentTemplate objects and all of their elements are collapsed into one and moved to the AdaptationSet. The playback device understands that each representation in this adaptation set uses this same template:

```
<MPD id="0" profiles="urn:dvb:dash:profile:dvb-dash:2014,urn:dvb:dash:profile:dvb-dash:isoff-ext-live:2014" type="dynamic" minimumUpdatePeriod="PT2S"
  minBufferTime="PT5S" timeShiftBufferDepth="PT1M0.002S"
  suggestedPresentationDelay="PT10S"
  availabilityStartTime="2024-01-01T00:00:00.000000+00:00" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xmlns="urn:mpeg:dash:schema:mpd:2011"
  xmlns:dvb="urn:dvb:dash-extensions:2014-1"
  xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/
ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
  publishTime="2025-02-27T19:52:52.000491+00:00">
  <ProgramInformation lang="eng" moreInformationURL="https://dvb.org/?standard=dvb-
mpeg-dash-profile-for-transport-of-iso-bmff-based-dvb-services-over-ip-based-networks">
    <Title>Test DVB-DASH Channel title</Title>
    <Source>Test DVB-DASH Channel source</Source>
    <Copyright>Test DVB-DASH Channel copyright</Copyright>
  </ProgramInformation>
  <BaseURL serviceLocation="slgroup" dvb:priority="1" dvb:weight="100">https://
w2dorn.egress.pentest-stack.dev.v2.mediapackage.elemental.aws.a2z.com/out/v1/
HippoPentest/HippoPentestCmaf/cmaf/</BaseURL>
```

```

<BaseURL serviceLocation="slgroup" dvb:priority="10" dvb:weight="10">https://
w2dorn.egress.pentest-stack.dev.v2.mediapackage.elemental.aws.a2z.com/out/v1/
HippoPentest/HippoPentestCmaf/cmaf/</BaseURL>
<Period id="1740685874005" start="PT10171H51M14.005S">
  <AdaptationSet id="107031859" contentType="video" mimeType="video/mp4"
segmentAlignment="true" startWithSAP="1" bitstreamSwitching="true">
    <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
    <SegmentTemplate timescale="60000" media="segment_${RepresentationID$_
$Number$.mp4" initialization="segment_${RepresentationID$_145057156_init.mp4"
startNumber="870342955" presentationTimeOffset="104441152310000">
      <SegmentTimeline>
        <S t="104441154540000" d="120000" r="29"/>
      </SegmentTimeline>
    </SegmentTemplate>
    <Representation id="h264_1080p30" bandwidth="4096000" codecs="avc1.4D4028"
frameRate="30/1" width="1920" height="1080"/>
    <Representation id="h264_720p30" bandwidth="8000000" codecs="avc1.4D401F"
frameRate="30/1" width="1280" height="720"/>
  </AdaptationSet>
  .
  .
  .
</Period>
<UTCTiming schemeIdUri="urn:mpeg:dash:utc:http-iso:2014" value="https://
time.akamai.com/?iso"/>
<Metrics metrics="DVBErrors">
  <Reporting schemeIdUri="urn:dvb:dash:reporting:2014" value="1"
dvb:reportingUrl="https://dvb.org" dvb:probability="1000"/>
  <Reporting schemeIdUri="urn:dvb:dash:reporting:2014" value="1"
dvb:reportingUrl="https://dvb.org" dvb:probability="500"/>
</Metrics>
</MPD>

```

Example full DASH manifest (uncompacted)

In the following example, the SegmentTemplate object and all of its elements are listed in every Representation. If ContentProtection is configured, those values are also listed in each Representation. Each adaptation set in the manifest has this same layout:

```

<AdaptationSet mimeType="video/mp4" segmentAlignment="true" subsegmentAlignment="true"
startWithSAP="1" subsegmentStartsWithSAP="1" bitstreamSwitching="true">
  <Representation id="1" width="640" height="360" frameRate="30/1" bandwidth="749952"
codecs="avc1.640029">

```

```

    <SegmentTemplate timescale="30000" media="index_video_1_0_$Number$.mp4?
m=1543947824" initialization="index_video_1_0_init.mp4?m=1543947824" startNumber="1">
      <SegmentTimeline>
        <S t="62000" d="60000" r="9"/>
      </SegmentTimeline>
    </SegmentTemplate>
  </Representation>
  <Representation id="2" width="854" height="480" frameRate="30/1" bandwidth="1000000"
codecs="avc1.640029">
    <SegmentTemplate timescale="30000" media="index_video_3_0_$Number$.mp4?
m=1543947824" initialization="index_video_3_0_init.mp4?m=1543947824" startNumber="1">
      <SegmentTimeline>
        <S t="62000" d="60000" r="9"/>
      </SegmentTimeline>
    </SegmentTemplate>
  </Representation>
  <Representation id="3" width="1280" height="720" frameRate="30/1"
bandwidth="2499968" codecs="avc1.640029">
    <SegmentTemplate timescale="30000" media="index_video_5_0_$Number$.mp4?
m=1543947824" initialization="index_video_5_0_init.mp4?m=1543947824" startNumber="1">
      <SegmentTimeline>
        <S t="62000" d="60000" r="9"/>
      </SegmentTimeline>
    </SegmentTemplate>
  </Representation>
</AdaptationSet>

```

DASH audio timeline pattern

In a DASH manifest, the `SegmentTimeline` element describes when each media segment is available and its duration. For audio tracks, the timescale mismatch between audio and video codecs means that audio segment durations don't always align evenly with video segment boundaries. This typically produces a repeating pattern of slightly different durations in the audio timeline. For example, an audio track might alternate between durations of 287232 and 288768 (at a timescale of 48000) in a predictable cycle.

Without patternization, the manifest must list every audio segment individually in the `SegmentTimeline`, using separate `S` elements for each duration change. As the time-shift buffer grows, the manifest grows linearly, which increases bandwidth usage and parsing time on playback devices.

When you enable the `PATTERNED` audio timeline setting, MediaPackage detects the repeating cycle of audio segment durations and represents it as a `Pattern` element in the manifest. Instead of listing each segment individually, the `SegmentTimeline` references the pattern by ID, and a single `S` element with a repeat count can describe many segments. Enabling this setting adds an `EssentialProperty` descriptor with the scheme `urn:mpeg:dash:pattern:2024` to the adaptation set. This descriptor signals to players that the manifest may contain patterned elements. The `EssentialProperty` is always present when the feature is enabled, but `Pattern` elements only appear when MediaPackage detects a qualifying repeating cycle in the audio segment durations. This follows the Segment Duration Patternization (SDP) specification from DASH-IF.

MediaPackage analyzes incoming audio segment durations and identifies repeating cycles (patterns of 2 to 6 segments). Once a pattern repeats at least twice, MediaPackage emits a `Pattern` element containing `P` sub-elements that describe the duration sequence. The `S` elements in the `SegmentTimeline` then reference this pattern using the `p` attribute (pattern ID) and the `pE` attribute (pattern entry point, indicating where in the cycle the segment begins).

If a segment arrives that doesn't match the expected pattern position, MediaPackage breaks from the pattern and represents that segment explicitly. Pattern detection then restarts. Similarly, when a discontinuity occurs (such as an input switch or stream restart), the pattern state resets and detection begins fresh on the new content.

When to use each setting

Choose between the two `AudioTimelinePattern` values based on your playback environment:

- **NONE** – The manifest uses an explicit per-segment timeline. Every audio segment duration is listed individually in the `SegmentTimeline`. Use this setting if your players don't support the DASH-IF SDP specification, or if you need maximum compatibility with older DASH clients.
- **PATTERNED** – The manifest includes the `urn:mpeg:dash:pattern:2024` `EssentialProperty` descriptor, and may contain `Pattern` elements when MediaPackage detects repeating duration cycles in the audio segments. When patterns are present, manifest size is significantly reduced for long time-shift buffers. Use this setting when your players support this feature and you want to minimize manifest bandwidth.

Caveats

Keep the following in mind when using audio timeline patterns:

- This feature applies only to audio timelines. Video and subtitle timelines are not patternized.
- When the PATTERNED setting is enabled, MediaPackage always includes the EssentialProperty descriptor `urn:mpeg:dash:pattern:2024` in the manifest, even if no pattern has been detected yet. If a player encounters this essential property, the manifest may contain Pattern elements. Players must support these elements to parse the manifest. Players that don't recognize it will reject the manifest.
- When a discontinuity occurs (for example, an input switch), the pattern resets. Segments around the discontinuity boundary are represented explicitly until a new pattern is detected.
- Uniform durations (where every segment has the same duration) are not patternized because the standard `S@r` repeat count already provides optimal compression for that case.

Example Unpatterned audio timeline

In this example, the audio AdaptationSet uses an explicit SegmentTimeline where every audio segment is listed individually. The durations alternate between 288768 and 287232, and the manifest grows linearly with the time-shift buffer:

```
<AdaptationSet id="1084563256" contentType="audio" mimeType="audio/mp4"
segmentAlignment="true" startWithSAP="1" bitstreamSwitching="true" lang="spa">
  <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
  <Label>spa</Label>
  <SegmentTemplate timescale="48000" media="segment_${RepresentationID}_${Number}.mp4"
initialization="segment_${RepresentationID}_295846802_init.mp4" startNumber="295918052"
presentationTimeOffset="85206900240000">
    <SegmentTimeline>
      <S t="85224398928384" d="288768"/>
      <S t="85224399217152" d="287232"/>
      <S t="85224399504384" d="288768"/>
      <S t="85224399793152" d="287232"/>
      <S t="85224400080384" d="288768"/>
      <S t="85224400369152" d="287232"/>
      <!-- ...additional alternating S elements for each segment in the buffer... -->
      <S t="85224426577152" d="287232"/>
      <S t="85224426864384" d="288768"/>
      <S t="85224427153152" d="287232"/>
    </SegmentTimeline>
  </SegmentTemplate>
  <Representation id="audio-ac3-spa" bandwidth="128000" codecs="ac-3"
audioSamplingRate="48000">
```

```

    <AudioChannelConfiguration
      schemeIdUri="tag:dolby.com,2014:dash:audio_channel_configuration:2011" value="A000"/>
    </Representation>
  </AdaptationSet>

```

Example Patterned audio timeline

The following example shows the same audio adaptation set with the PATTERNED setting enabled. The repeating duration cycle is defined once in a Pattern element, and a single S entry with a repeat count covers most of the timeline. A trailing S entry handles the final segment that doesn't complete a full pattern cycle:

```

<AdaptationSet id="1084563256" contentType="audio" mimeType="audio/mp4"
  segmentAlignment="true" startWithSAP="1" bitstreamSwitching="true" lang="spa">
  <EssentialProperty schemeIdUri="urn:mpeg:dash:pattern:2024"/>
  <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main"/>
  <Label>spa</Label>
  <SegmentTemplate timescale="48000" media="segment_${RepresentationID}_${Number}.mp4"
    initialization="segment_${RepresentationID}_295846802_init.mp4" startNumber="295918052"
    presentationTimeOffset="85206900240000">
    <Pattern id="008c49c1">
      <P d="287232"/>
      <P d="288768"/>
    </Pattern>
    <SegmentTimeline>
      <S t="85224398928384" r="49" p="008c49c1" pE="1"/>
      <S t="85224427153152" d="287232"/>
    </SegmentTimeline>
  </SegmentTemplate>
  <Representation id="audio-ac3-spa" bandwidth="128000" codecs="ac-3"
    audioSamplingRate="48000">
    <AudioChannelConfiguration
      schemeIdUri="tag:dolby.com,2014:dash:audio_channel_configuration:2011" value="A000"/>
    </Representation>
  </AdaptationSet>

```

When you enable the PATTERNED setting, MediaPackage adds the EssentialProperty descriptor with schemeIdUri="urn:mpeg:dash:pattern:2024" at the AdaptationSet level. This descriptor signals to DASH players that the adaptation set may contain Pattern elements. You will never encounter pattern elements without this descriptor present, but the descriptor may be present without any pattern elements if no repeating cycle has been detected. Players that

don't recognize this scheme are required by the DASH specification to ignore the adaptation set, which prevents misinterpretation of any Pattern elements that may be present. For this reason, only enable PATTERNED when your target players support `urn:mpeg:dash:pattern:2024`.

To configure the audio timeline pattern for a DASH endpoint, set the `AudioTimelinePattern` field in the [manifest configuration](#).

HLS and LL-HLS in AWS Elemental MediaPackage

MediaPackage supports HTTP Live Streaming (HLS) and Low-Latency HLS (LL-HLS) for delivering content to a wide range of devices, particularly Apple devices and browsers. HLS is one of the most widely supported adaptive bitrate streaming protocols, while LL-HLS extends the standard HLS protocol to provide reduced latency for near real-time streaming experiences.

HLS and LL-HLS features and capabilities

When using HLS or LL-HLS in MediaPackage, you can take advantage of the following features:

- **Container types:** HLS and LL-HLS support both TS and CMAF container types, providing flexibility for different use cases.
- **Program date/time:** MediaPackage can insert `EXT-X-PROGRAM-DATE-TIME` tags at configurable intervals, enabling features like viewer seek in the playback timeline and time display on the player.
- **Start tag:** MediaPackage supports adding `EXT-X-START` tags to control where playback begins in the manifest.
- **I-frame only streams:** MediaPackage can include additional I-frame only streams to enable player functionality like fast forward and rewind.
- **Audio rendition groups:** For TS containers, MediaPackage can group all audio tracks into a single rendition group, allowing all other tracks to be used with any audio rendition.
- **SCTE-35 ad markers:** MediaPackage supports signaling ad markers in HLS manifests using Daterange format or SCTE-35 enhanced CUE tags.
- **Encryption:** HLS supports AES-128 and Sample AES encryption methods with FairPlay (Sample AES only) and Clear Key AES-128 (AES-128 only) DRM systems.

HLS manifest structure

An HLS manifest is a text-based M3U8 playlist that describes the available media segments and their characteristics. HLS uses a hierarchical structure with two types of playlists:

- **Multivariant playlist:** Contains information about the available variants (different bitrates and resolutions) and references to the media playlists.
- **Media playlist:** Contains references to the actual media segments for a specific variant.

Key tags in HLS manifests include:

- **EXT-X-STREAM-INF:** Describes a variant stream in the multivariant playlist.
- **EXT-X-MEDIA:** Describes alternative renditions such as audio or subtitles.
- **EXTINF:** Specifies the duration of a media segment.
- **EXT-X-PROGRAM-DATE-TIME:** Associates a segment with an absolute date and time.
- **EXT-X-KEY:** Specifies how to decrypt encrypted segments.
- **EXT-X-SESSION-KEY:** Provides encryption key information in the multivariant playlist to enable client key pre-fetching for DRM-protected content.
- **EXT-X-DATERANGE:** Signals ad markers and program transition events.

LL-HLS extends the standard HLS protocol with additional tags and features to reduce latency:

- **EXT-X-PART:** Identifies partial segments that can be delivered before a complete segment is available.
- **EXT-X-PRELOAD-HINT:** Provides hints to the player about upcoming content.
- **EXT-X-SERVER-CONTROL:** Specifies server parameters for low-latency streaming.

Differences between HLS and LL-HLS

While HLS and LL-HLS share many features, there are important differences to consider:

- **Latency:** Standard HLS typically has latency of 18-30 seconds, while LL-HLS can achieve latency as low as 3-5 seconds.
- **Program date/time:** Program date/time (PDT) is optional for standard HLS but required for LL-HLS.

- **Segment delivery:** LL-HLS delivers partial segments to reduce latency, while standard HLS delivers complete segments.
- **Player compatibility:** LL-HLS requires players that specifically support the low-latency extensions.

When choosing between HLS and LL-HLS, consider your latency requirements, player compatibility, and the importance of features like trick play (fast forward/rewind).

HLS and LL-HLS use cases

Consider using HLS or LL-HLS endpoints in the following scenarios:

- Delivering content to Apple devices (iPhone, iPad, Apple TV) and browsers
- Implementing low-latency streaming for live events or interactive experiences (LL-HLS)
- Supporting DRM-protected content with FairPlay or AES-128 encryption
- Creating multi-protocol streaming workflows that need to include HLS alongside other formats
- Enabling advanced playback features like fast forward and rewind with I-frame only streams

For information about creating HLS or LL-HLS endpoints, see [Create an HLS or LL-HLS manifest](#).

Dynamic Multiview in AWS Elemental MediaPackage

Dynamic Multiview is a server-side capability that composes multiple live video sources into a single encoded video stream. Content providers can deliver multi-angle, multi-game, and personalized viewing experiences as standard HLS and DASH streams playable on most modern consumer devices, televisions, and set top boxes without custom player development.

Dynamic Multiview provides the following benefits:

- **Plays on the devices you already reach.** The output is one video track in H.264 (AVC) or H.265 (HEVC), delivered over HLS or DASH. Any device that can play your existing streams can play a multiview stream — there is no custom player, no client-side compositing, and no additional SDK. A single DRM key covers every layout.
- **Viewers keep control of audio and captions.** Every view's audio and caption tracks are carried in the manifest as selectable renditions, and a viewer switches between them without changing the visual layout.

- **Views stay synchronized.** Sources are delivered as time-aligned segments, so all views in an assembled frame remain in sync.
- **Encoding cost scales with sources, not combinations.** You encode one ABR stack for each feed no matter how many arrangements viewers request. Combinations are assembled at the point of delivery rather than pre-rendered.

Dynamic Multiview is a capability of AWS Elemental MediaLive and MediaPackage. MediaLive encodes each source as its own channel with encoding constraints that make the output combinable downstream. These feeds are ingested into MediaPackage as individual source channels, and MediaPackage assembles them into whatever layout a viewer requests, on demand.

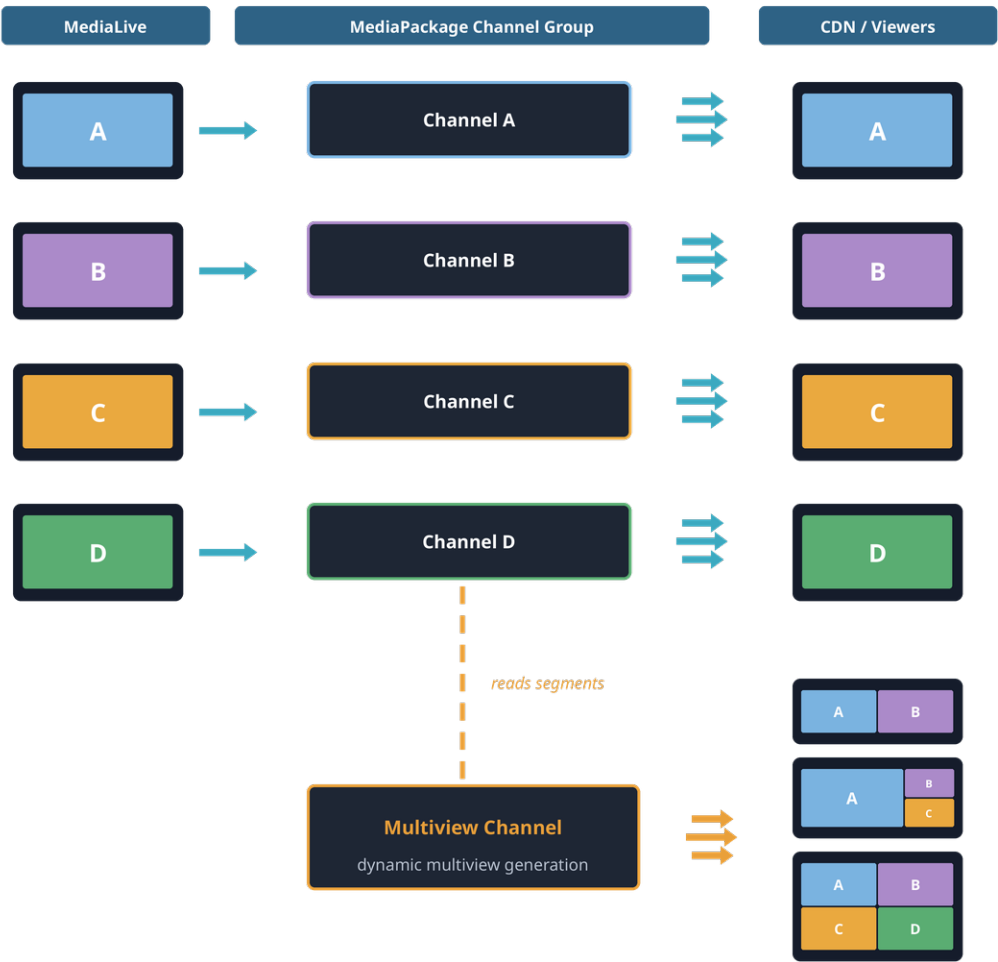
MediaPackage multiview channels

On MediaPackage, a multiview workflow uses two kinds of channels:

- **Source channels** — ordinary CMAF channels, each receiving one camera angle or event from your encoder. Each source provides the content for one view in the assembled layout.
- **A multiview channel** — a channel with an input type of MULTIVIEW. It receives no ingest of its own. Instead it declares which source channels it can use, and which layouts viewers can request. You create origin endpoints on the multiview channel, and viewers play back from those endpoints.

In this guide, a **source** is a channel that feeds content into the multiview channel, and a **view** is one region of the composited frame. Each view is normally filled by a source; if that source is unavailable, the view shows black.

All source channels and the multiview channel must be in the same channel group. For more information about channel groups, see [the section called “Working with channel groups”](#).



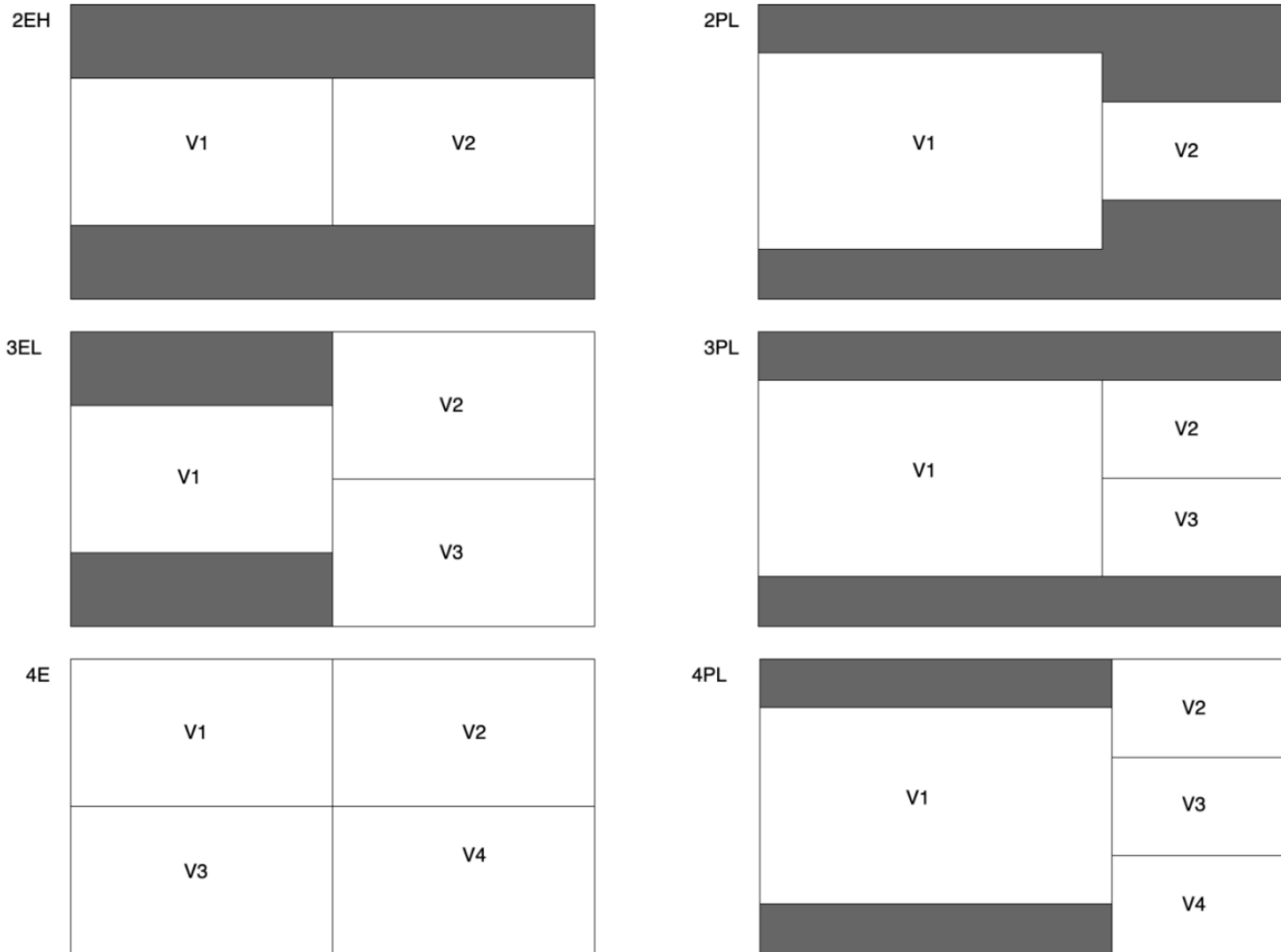
Layouts

A layout defines how many views are composited and how they are arranged. Each layout accepts an exact number of sources.

Dynamic Multiview layouts

Layout	Views	Arrangement	View sizes
2EH	2	Two views side by side in the middle of the frame, with full-width bars above and below	Equal

Layout	Views	Arrangement	View sizes
3EL	3	One view in the left column, with black bars above and below it, beside two stacked views on the right	Equal
4E	4	Four equal views in a 2×2 grid	Equal
2PL	2	One large primary view on the left, one smaller view on the right, with full-width bars above and below	Primary is twice the width and height of the secondary
3PL	3	One large primary view on the left, two stacked views on the right, with full-width bars above and below	Primary is twice the width and height of each secondary
4PL	4	One large primary view on the left, with black bars above and below it, and three stacked views on the right	Primary is twice the width and height of each secondary



In 2PL, 3PL, and 4PL, one view is larger than the others. This large primary view is V1, filled by the first source you list; the remaining sources fill the smaller views (V2, V3, and V4) in the order you list them.

Black areas in the assembled frame

An assembled multiview frame contains black in two places, from two different causes:

- **Layout padding.** Every layout except 4E places synthetic black fill around the views so the combined frame keeps a standard 16:9 aspect ratio. 4E composites its four views edge to edge with no padding.
- **Per-view borders.** MediaLive adds a black border to each participating encode so that video compression artifacts from one view cannot bleed into its neighbor. MediaPackage does not trim these, so the black a viewer sees *between* two views is the sum of the two adjacent borders, while

the outer edge of the frame shows only one. Border thickness is proportional to rendition size and is configured in MediaLive. For more information about configuring border thickness, see [Dynamic Multiview: Borders](#) in the MediaLive user guide.

Requirements

Encoder. Source feeds must be encoded by AWS Elemental MediaLive. Multiview compositing relies on encoder output that only MediaLive produces; third-party encoders are not supported. MediaLive's encoder-side requirements are documented in the MediaLive user guide. Configure those first; this guide covers the MediaPackage side.

- Overall requirements and validation: see [Dynamic Multiview: Requirements and Limitations](#).
- Per-output encoder settings and dimension rules: see [Dynamic Multiview: MediaLive Configuration Reference](#).
- Borders: see [Dynamic Multiview: Borders](#).

Source channels. Each source channel must:

- use the **CMAF** input type
- be in the same channel group as the multiview channel
- not itself be a MULTIVIEW channel
- use the **epoch-locked** output locking mode

The multiview channel must also be epoch-locked. Output locking mode is fixed when a channel is created, so existing non-epoch-locked CMAF channels cannot be used as multiview sources (see [Output locking mode](#)). Input type is likewise immutable after creation: you cannot convert a CMAF channel into a multiview channel.

Codecs. Participating sources must be H.264 (AVC) or H.265 (HEVC). Both codecs can exist in the same workflow, but MediaPackage combines only sources encoded with the **same** codec — you can't combine an AVC source and an HEVC source in the same multiview. Frame rates must match across the sources you combine. AV1 is not supported for multiview.

HLS child manifest URL encoding. On multiview endpoints, the `UrlEncodeChildManifest` setting on each HLS manifest defaults to `true`, and you should keep it set to `true`. Multiview requires this for HLS playback: it URL-encodes the `aws.multiview` query string that

MediaPackage embeds in the child manifest URLs, so that HLS players request the child manifests correctly. Setting it to `false` (the previous default) causes HLS playback to fail on players that request the child manifest URL as-is.

Set up a multiview workflow

1. **Create the source channels.** Create one CMAF, epoch-locked channel per camera angle or event and point your MediaLive channel at its ingest endpoints. See [the section called “Creating a channel”](#).
2. **Create the multiview channel.** Create a channel with an input type of `MULTIVIEW` and a multiview configuration that declares:
 - `AvailableSources` — the names of the source channels, in the same channel group, that the multiview channel can use.
 - `AvailableLayouts` — the layouts that viewers may request. At least one.

```
aws mediapackagev2 create-channel \  
  --channel-group-name exampleChannelGroup \  
  --channel-name exampleMultiviewChannel \  
  --input-type MULTIVIEW \  
  --multiview-configuration '{  
    "AvailableSources": ["cam1", "cam2", "cam3", "cam4"],  
    "AvailableLayouts": ["LAYOUT_4E", "LAYOUT_3PL", "LAYOUT_2EH"]  
  }'
```

The multiview channel is added to each source channel's `AttachedMultiviewChannels` property. This field is read-only, and a source channel cannot be deleted while a multiview channel still lists it as an available source. To delete a source channel, first remove it from the multiview channel's `AvailableSources`.

3. **Create an origin endpoint** on the multiview channel, as described in [Creating an origin endpoint](#). Multiview endpoints serve HLS from CMAF or TS containers, and DASH from CMAF containers.

For an end-to-end walkthrough of creating channels and endpoints in MediaPackage, see [Getting started](#).

Request a multiview stream

Viewers select a view arrangement by adding the `aws.multiview` query parameter to the playback URL:

```
aws.multiview=layout:<layout>;sources:<source1>,<source2>,...
```

- `layout` — one of the layouts declared in the channel's `AvailableLayouts`.
- `sources` — a comma-separated list of source channel names. The number of sources must exactly match the layout's view count.

The order of sources determines which view each source fills: the first source you list appears in V1, the second in V2, the third in V3, and the fourth in V4, as labeled in the layout diagrams in [the section called “Layouts”](#). MediaPackage never reorders the list, so two requests that name the same sources in a different order are different streams and produce different manifests.

Layout values in the playback URL omit the `LAYOUT_` prefix. A channel configured with `LAYOUT_4E` is played back with `layout:4E`; the prefixed form `layout:LAYOUT_4E` is rejected.

The parameter must be URL-encoded in the playback URL:

```
https://<egress-domain>/out/v1/exampleChannelGroup/exampleMultiviewChannel/hls/index.m3u8?aws.multiview=layout%3A4E%3Bsources%3Acam1%2Ccam2%2Ccam3%2Ccam4
```

MediaPackage propagates the parameter into the child manifest and segment URLs it emits, so a player only needs to add it to the top-level manifest request.

The same source channel may appear more than once in `sources`.

Audio, captions, and source loss

MediaPackage carries an audio track and a caption track for each source channel included in the multiview, as selectable renditions in the manifest, so a viewer can switch audio or captions without changing the visual layout. Supported captions are WebVTT and TTML.

When a source is unavailable, MediaPackage keeps the stream continuous by substituting for the missing content: black video for that source's view, silent audio for its audio track, and empty captions for its caption track.

Manifest filtering is supported on multiview endpoints, including language pruning and bitrate capping. For more information, see [Manifest filtering](#).

Identify a view's audio and captions

MediaPackage labels the audio and caption renditions in a stitched manifest so that you, or a player, can tell which view each rendition belongs to. The label comes from the name you give the track in MediaLive, so name each audio and caption track that participates in the multiview.

Name the tracks in MediaLive. In your MediaLive channel:

- For audio, set the **Stream Name** on each audio description — for example, English or Spanish.
- For captions, set the **Language description** on each caption description. Caption descriptions don't have a stream name; the **Language description** is the equivalent field.

The view prefix that MediaPackage adds makes each rendition name unique, even when two source channels use the same track name. Choose names that describe the content — the language, for example — so that a viewer can tell the renditions apart. For the MediaLive output requirements, see [Dynamic Multiview: Requirements and Limitations](#). For where these fields appear in the MediaLive console, see [How Dynamic Multiview works](#), both in the MediaLive user guide.

HLS. MediaPackage prefixes the name you set with the view position and uses the result as the NAME attribute on the EXT-X-MEDIA tag. An audio track whose stream name is English, filling the first view, appears as View1_English:

```
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio_0",NAME="View1_English",URI="..."
```

Captions follow the same pattern, using the language description. A caption track whose language description is English and that belongs to the second view appears as View2_English on its TYPE=SUBTITLES tag. To find a view's audio or captions in an HLS manifest, look for the View prefix on the NAME attribute.

DASH. MediaPackage puts the same view-prefixed name in the Label element of the audio or subtitle adaptation set — for example View1_English — mirroring the HLS NAME. Each audio and subtitle adaptation set also carries a Viewpoint descriptor whose value identifies the view — View1, View2, and so on. This value matches the corresponding view's video, so a player can associate a track with its view programmatically, independent of the label text.

In both formats, the view number matches the order in which you listed the source channels in the `sources` parameter of your request: the first source is `View1`, the second is `View2`, and so on. For more information, see [Request a multiview stream](#).

Ad signaling (SCTE-35)

Dynamic Multiview supports full-screen ad replacement signaled by SCTE-35 from the primary view. MediaPackage processes SCTE-35 messages from the primary view only. SCTE-35 messages delivered inside segments are not supported. For more information about SCTE-35 handling in MediaPackage, see [the section called “SCTE-35 messages”](#).

CDN caching

Because the view arrangement lives in the query string, each distinct combination of layout and source order is a distinct object. Configure your CDN to include the `aws.multiview` query parameter in the cache key, so that viewers requesting different arrangements are not served each other's content. If you expect many combinations, consider constraining the arrangements your application offers, so that your CDN cache stays effective.

Constraints

Playback and manifests

- Low-latency HLS is not supported.
- Microsoft Smooth Streaming is not supported, and a multiview endpoint cannot use the ISM container.
- Harvest jobs and live-to-VOD are not supported on multiview endpoints.
- Time-shifted playback is not supported: the `start` and `end` parameters, clip start time, the startover window, and `EXT-X-START` cannot be used.
- Time delay is supported up to a maximum of 24 hours.
- The manifest window cannot exceed 900 seconds (15 minutes).
- I-frame-only playlists and trickplay tracks are not supported.

Content protection

- A single DRM key covers every view and every layout. Per-view DRM keys are not supported.

For more information about encryption and DRM, see [the section called “Content encryption and DRM”](#).

Channel operations

- Input switching based on MQCS and MQCS publishing in CMSD are not available on multiview channels.

Manifest filtering from AWS Elemental MediaPackage

With manifest filtering, AWS Elemental MediaPackage dynamically produces client manifests based on filter parameters that you specify. This enables you to do things such as restrict viewer access to premium 4K HEVC content, or target specific device types and audio sample rate ranges, all from a single endpoint.

With manifest filtering, the resulting manifest from MediaPackage includes only the audio and video streams that match the characteristics that you specify in your filters. If no manifest filter is used, then all of the ingested streams are present in the endpoint output stream. The exception to this is if you have set stream filters for the endpoint, such as minimum video bitrate. In that case, the manifest filter is applied after the stream filter, which could skew your output, and is not recommended.

Manifest filtering can be used on all origin endpoint types supported by MediaPackage.

To use manifest filtering, you can do one of the following:

- Apply filters to all manifests that originate from an endpoint: Set the filters on the origin endpoint through the MediaPackage console or API. For console instructions, see [Creating an origin endpoint](#).
- Apply filters dynamically across requests: Ensure that downstream video players include the appropriate query parameters in their manifest requests. For information about using query parameters, see [Manifest filtering query parameters](#).

Note

When configuring Manifest filters as part of the Filter Configuration for your endpoint, be aware that using corresponding query parameters in the manifest's endpoint URL can lead

to issues. Specifically, if you include a Manifest filter in your Filter Configuration and then attempt to use matching query parameters in the endpoint URL, you will encounter a 404 HTTP error code. For example, if your Filter Configuration includes a Manifest filter with an `audio_sample_rate` key set to 44100, and you subsequently make an HTTP request to `https://<example-url>/?aws.manifestfilter=audio_sample_rate:44100` this will result in a 404 error. The Filter Configuration applies universally to all egress requests for your endpoint, and duplicating these settings in the URL query can lead to conflicts and errors. For more information about Manifest filters, see [Manifest filtering from AWS Elemental MediaPackage](#).

The following topics describe how manifest filtering works with MediaPackage.

Topics

- [Manifest filtering query parameters in AWS Elemental MediaPackage](#)
- [Special conditions for TS and CMAF manifests in MediaPackage](#)
- [Manifest filtering examples in AWS Elemental MediaPackage](#)
- [AWS Elemental MediaPackage manifest filtering error conditions](#)

Manifest filtering query parameters in AWS Elemental MediaPackage

To use manifest filtering query parameters, append `aws.manifestfilter` to your playback request to MediaPackage. MediaPackage evaluates the query, and serves a client manifest based on those query parameters.

The following sections describe how to configure manifest filtering query parameters.

Note

If you are using TS or CMAF origin endpoints, special conditions apply. For information about these conditions, see [Special conditions for TS and CMAF manifests in MediaPackage](#).

Query parameter formatting

Use the following guidelines when constructing query parameters:

- Queries are not case sensitive.
- Queries can be up to 1,024 characters.
- Reserved characters in the queries must be URL encoded as indicated in the [URI: General Syntax](#) standard.

Important

At a minimum, if your query includes multiple parameters, the following characters in the query must be URL encoded. If they're not, MediaPackage processes just the first query parameter in the string.

Character	Encoded value
: (colon)	%3A
; (semicolon)	%3B
, (comma)	%2C
+ (plus)	%2B

If the query is malformed, or if there aren't streams that match the query parameters, MediaPackage returns an incomplete or empty manifest. For query syntax, see the following section.

Query syntax

The base query parameter is `aws.manifestfilter`, which is followed by optional parameter name and value pairs. To construct the query, append `?aws.manifestfilter=` to the end of the MediaPackage endpoint URL, followed by parameter names and values. For a list of all of the available parameters, see [Manifest filtering query parameters in AWS Elemental MediaPackage](#).

Example HLS filter query

The following example query includes filters for audio sample rate, video bitrate, video codec, and audio language. Note that the reserved characters use URL-encoded values.

`https://example-mediapackage-endpoint.mediapackage.us-west-2.amazonaws.com/out/v1/examplemediapackage/index.m3u8?aws.manifestfilter=audio_sample_rate%3A0-44100%3Bvideo_bitrate%3A0-2147483647%3Bvideo_codec%3Ah265%3BAudio_language%3Afr%2Cen-US%2Cde`

The query syntax is described in the following table.

Note

If you use Amazon CloudFront as your CDN, you might need to set additional configurations. For more information, see [Configure cache behavior for all endpoints](#).

Query string component	Description
<code>?</code>	A restricted character that marks the beginning of a query.
<code>aws.manifestfilter</code> <code>=</code>	The base query, which is followed by parameters constructed of name and value pairs. For a list of all of the available parameters, see Manifest filtering query parameters in AWS Elemental MediaPackage .
<code>:</code>	Used to associate the parameter name with a value. For example, <i>parameter_name :value</i> . When using multiple query parameters, this character must be URL encoded (%3A).
<code>;</code>	Separates parameters in a query that contains multiple parameters. For example, <i>parameter1_name:value ;parameter2_name:minValue-maxValue</i> . When used in a list of parameters for the same query, implies an AND operation. When using multiple query parameters, this character must be URL encoded (%3B).
<code>,</code>	Separates a list of values. For example, <i>parameter_name: value1,value2,value3</i> . Comma-separated values in a list imply an OR relationship.

Query string component	Description
	When using multiple query parameters, this character must be URL encoded (%2C).
-	Used to define a parameter's minimum - maximum value range. For example, <code>audio_sample_rate:0-44100</code> . When a numerical value is used in a range, it is included in the range definition. This means that streams must be greater than or equal to the minimum value, and less than or equal to the maximum value. With ranges, the minimum and maximum values are mandatory. The supported range values are <code>0 - 2147483647</code> .

Query value formats

The following query parameters support expanded value formats:

- `audio_bitrate`
- `audio_channels`
- `audio_sample_rate`
- `trickplay_height`
- `video_bitrate`
- `video_framerate`
- `video_height`

For all of these parameters, you can format your values as single values or ranges, one or more ranges, or a combination of both.

Individual values and ranges

Filter manifests by single values or ranges.

Syntax

- Individual single values: `aws.manifestfilter=parameter:value`
- Individual range: `aws.manifestfilter=parameter:min1-max1`

Example individual value

The following example filters for videos with a bitrate of 8000000 bps.

```
stream.mpd?aws.manifestfilter=video_bitrate:8000000
```

Note

When you filter for a single `video_framerate` value, MediaPackage uses an approximate equals comparison with an epsilon tolerance of 0.0005. MediaPackage uses this tolerance because the query allows only up to three decimal places, and there could be a small difference in accuracy between the stored framerates and the specified framerate.

For example, if your filter is `video_framerate:30.000`, MediaPackage matches framerates in the range of 29.9995 to 30.0005.

Multiple values and ranges

Filter manifests by multiple single values or multiple ranges.

Syntax

- Multiple single values: `aws.manifestfilter=parameter:value,value`
- Multiple ranges: `aws.manifestfilter=parameter:min1-max1,min2-max2`

Example multiple ranges

The following example filters for videos that are either 240p-360p OR 720p-1080p. It uses URL encoding for reserved characters.

```
stream.mpd?aws.manifestfilter=video_height%3A240-360%2C720-1080
```

Combination

Filter manifests by a combination of ranges and values.

Syntax

- Multiple ranges and single values: `aws.manifestfilter=parameter:min1-max1,min2-max2,value1,value2`

Example multiple ranges and values

The following example filters for videos that are either 240p-360p, 720p-1080p, 1440p, OR 2160p. It uses URL encoding for reserved characters.

```
stream.mpd?aws.manifestfilter=video_height%3A240-360%2C720-1080%2C1440%2C2160
```

Query parameters

MediaPackage supports the following query parameters.

You can set one or more filters. For example, to restrict all manifests from this endpoint to 0 to 44000 Hz audio sample rate, 0 to 2147483647 video bitrate, H265 video codec, and French and English languages, enter the following key and value pairs on the origin endpoint:

Filter key `audio_sample_rate` | **Filter value:** 0-44100

Filter key `video_bitrate` | **Filter value:** 0-2147483647

Filter key `video_codec` | **Filter value:** H265

Filter key `audio_language` | **Filter value:** fr,en-US

Category	Name	Description	Example
Audio	<code>audio_bitrate</code>	- The audio bitrate in bits per second. Accepted values: Two integers aggregated with a dash that define an inclusive range. The supported range values are 0 - 2147483647 . Individual integers within that range are also allowed.	<code>stream.mpd?aws.manifestfilter=audio_bitrate:0-2147483647</code>
Audio	<code>audio_channels</code>	- The number of audio channels. Accepted values: Two integers aggregated with a dash that define an inclusive range. The supported range values are 1 - 32767. Individual	<code>stream.mpd?aws.manifestfilter=audio_channels:1-8</code>

Category	Name	Description	Example
		l integers within that range are also allowed.	
Audio	audio_codec	- The audio codec type. Accepted values: AACL, AACH, AC-3, EC-3. You must include the - for AC-3 and EC-3. The values are <i>not</i> case-sensitive.	stream.mpd?aws.manifestfilter=audio_codec:AACL,AC-3
Audio	audio_language	- Audio languages or functional codes derived from encoder passthrough. Accepted values: Arbitrary strings, such as two or four character ISO-639-1 language codes. You must use the same language strings that are set for your encoder. The values are <i>not</i> case-sensitive.	stream.mpd?aws.manifestfilter=audio_language:fr,en-US,de
Audio	audio_sample_rate	- The audio sample rate in Hz. Accepted values: Two integers aggregated with a dash that define an inclusive range. The supported range values are 0 - 2147483647 . Individual integers within that range are also allowed.	stream.mpd?aws.manifestfilter=audio_sample_rate:0-44100

Category	Name	Description	Example
Subtitle	subtitle_language	- The subtitle language or functional codes derived from encoder passthrough. Accepted values: Arbitrary strings, such as two or four character ISO-639-1 language codes. You must use the same language strings that are set for your encoder. The values are <i>not</i> case-sensitive.	stream.mpd?aws.manifestfilter=subtitle_language:en-US,hi
Video	trickplay_height	- The height of the trick-play image in pixels. This applies to both I-frame only and image-based trick-play. Note If you're using this parameter with I-frame only trick-play, <code>trickplay_height</code> and <code>video_height</code> should have similar values. If the values are not the same, I-frame only tracks might be removed from a manifest. Accepted values: Two integers aggregated with a dash that define an inclusive range. The supported range values are 1 - 2147483647 . Individual integers within that range are also allowed.	stream.mpd?aws.manifestfilter=trickplay_height:200-1200

Category	Name	Description	Example
Video	trickplay_type	- The trickplay track type. Accepted values: iframe, image, none. The values are <i>not</i> case-sensitive.	<code>stream.mpd?aws.manifestfilter=trickplay_type:iframe</code>

Category	Name	Description	Example
Video	video_bitrate	- The video bitrate in bits per second. Note If you're using this parameter, we recommend that you use only the <code>video_bitrate</code> filter parameter to set the video bitrate. Don't also set the minimum and maximum video bitrate via the MediaPackage console or AWS CLI. The <code>video_bitrate</code> filter applies to the video bitrate settings created at the endpoint. If you use the parameter and set the bitrate in the console or AWS CLI, your output might be skewed. Accepted values: Two integers aggregated with a dash that define an inclusive range. The supported range values are 0 - 2147483647. Individual integers within that range are also allowed. Note You can't use this parameter with trick-play streams.	<pre>stream.mpd?aws.manifestfilter=video_bitrate:0-2147483647</pre>

Category	Name	Description	Example
Video	video_codec	- The video codec type. Accepted values: H264, H265, AV1. The values are <i>not</i> case-sensitive.	stream.mpd?aws.manifestfilter=video_codec:h264
Video	video_dynamic_range	- The video dynamic range. Accepted values: dv, hdr10, hlg, sdr. The values are <i>not</i> case-sensitive. You can use the dv value to filter for Dolby Vision streams.	stream.mpd?aws.manifestfilter=video_dynamic_range:hdr10
Video	video_framerate	- The video frame rate range in the NTSC format. Accepted values: Two floating-point numbers aggregated with a dash that define an inclusive range. Each number can have up to three optional fractional values. For example, 29.97 or 29.764. The supported range values are 1 - 999.999. Individual integers within that range are also allowed.	stream.mpd?aws.manifestfilter=video_framerate:23.976-30

Category	Name	Description	Example
Video	video_height	- The height of the video in pixels.  Note If you're using this parameter with I-frame only trick-play, <code>trickplay_height</code> and <code>video_height</code> should have similar values. If the values are not the same, I-frame only tracks might be removed from a manifest. Accepted values: Two integers aggregated with a dash that define an inclusive range. The supported range values are 1 - 32767. Individual integers within that range are also allowed.	<pre>stream.mpd?aws.manifestfilter=video_height:720-1080</pre>

Category	Name	Description	Example
DRM	<code>aws.drmsettings</code>	- Controls DRM-related settings for HLS manifests. Currently supports excluding session keys from the multivariant playlist. Accepted values: <code>exclude_session_keys</code> - Removes <code>EXT-X-SESSION-KEY</code> tags from the multivariant playlist while preserving key information in individual media playlists. Use case: Useful for legacy clients that have issues with session key pre-fetching or when using manifest filtering to control access to specific content variants. Scope: Only applies to HLS manifests. Ignored for DASH manifests.	<code>stream.m3u8?aws.drmsettings=exclude_session_keys</code>

Special conditions for TS and CMAF manifests in MediaPackage

If you are using TS or CMAF manifests for MediaPackage, these special conditions apply.

- For TS manifests, we strongly recommend that you use audio rendition groups to avoid removing the video streams that are multiplexed with the audio streams that are filtered out. For more information about rendition groups, see [AWS Elemental MediaPackage rendition groups reference](#).
- In TS and CMAF manifests, the audio sample rate is not signaled, so it's not easy to visually check the original or filtered manifests for this setting. To verify the audio sample rate, check the audio sample rate at the encoder level and output level.
- In TS and CMAF manifests, the `BANDWIDTH` attribute for a variant associates the bandwidth of the audio track with the video track, whether it is multiplexed with the video track, or if it is an audio rendition track referenced by the video track. Therefore, you can't visually inspect the

original and filtered manifests to confirm the `video_bitrate` filter has worked. To verify the filter, check the video bitrate at the encoder level and output level.

- For TS and CMAF manifests, request parameters appended to bitrate playlists or segments result in an HTTP 400 error.

Manifest filtering examples in AWS Elemental MediaPackage

These are MediaPackage manifest filtering examples.

Example 1: Include only videos with specific heights

You want to include only videos with heights that the most popular playback devices support. You set the `video_height` to filter out video streams that don't meet one of these height requirements: 240p through 360p, 720p through 1080p, 1440, or 2160p.

```
?aws.manifestfilter=video_height%3A240-360%2C720-1080%2C1440%2C2160
```

Example 2: Target a player that supports AVC and a 44.1k audio sample rate

The viewer is playing content on a device that can only support AVC and a 44.1k audio sample rate. You set the `video_codec` and `audio_sample_rate` to filter out streams that don't fit these requirements.

```
?aws.manifestfilter=audio_sample_rate%3A0-44100%3Bvideo_codec%3Ah264
```

Example 3: Restrict 4k HEVC content

Your 4K HEVC stream is 15 Mbps, and all your other streams are less than 9 Mbps. To exclude the 4K stream from the stream set, you set a threshold of 9,000,000 bits per second to filter out the higher bitrate.

```
?aws.manifestfilter=video_bitrate:0-9000000
```

Example 4: Include video between 23.976 and 30 frames per second

To only include video within a certain frame rate range, use `video_framerate`. This parameter accepts floating-point numbers with up to three optional decimal values.

```
?aws.manifestfilter=video_framerate:23.976-30
```

AWS Elemental MediaPackage manifest filtering error conditions

Common error conditions for manifest filtering with MediaPackage are listed in the following table.

Error condition	Example	HTTP status code
A list parameter is not found and is not part of a constrained list	?aws.manifestfilter=audio_language:da ahlia	200
Only subtitle streams are present in the stream	?aws.manifestfilter=audio_sample_rate:0-1;video_bitrate=0-1	200
Duplicate filter parameter	?aws.manifestfilter=audio_sample_rate:0-48000;aws.manifestfilter=audio_sample_rate:0-48000	400
Invalid parameter	?aws.manifestfilter=donut_type:rhododendron	400
Invalid range parameter	?aws.manifestfilter=audio_sample_rate:300-0	400
Invalid range value (more than INT_MAX)	?aws.manifestfilter=audio_sample_rate:0-2147483648	400
Malformed query string	?aws.manifestfilter=audio_sample_rate:is:0-44100	400

Error condition	Example	HTTP status code
Parameter string is greater than 1024 characters	?aws.manifestfilter=audio_language:abcdefghijklmnopqrstuvwxyz....	400
Query parameters on an TS or CMAF bitrate manifest	index_1.m3u8?aws.manifestfilter=video_codec:h264	400
Query parameters on a segment request	..._1.[ts mp4 vtt.].?aws.manifestfilter=video_codec:h264	400
Repeated query parameter	?aws.manifestfilter=audio_sample_rate:0-48000;aws.manifestfilter=video_bitrate:0-1	400
Application of the filter results in an empty manifest (content has no streams that meet the conditions defined in the query string)	?aws.manifestfilter=audio_sample_rate:0-1;video_bitrate=0-1	400

Leveraging media quality scores with AWS Elemental MediaPackage

Streaming media quality is influenced by many factors. To ensure the highest quality viewing experience, services in the video streaming workflow must have awareness of, and ability to act on, indicators of quality down to the segment level in a stream. MediaPackage uses media quality confidence score (MQCS) functionality to receive, act on, calculate, and communicate quality scores of live streaming content. When adopted into your entire workflow, MQCS provides an automated operation to ensure that you provide the optimum viewing experience to your customers.

The following sections describe how MediaPackage leverages media quality scores from stream inputs and through origin outputs to continuously monitor video quality, correct for errors in the live streaming workflow, and help facilitate debugging through increased visibility.

MQCS is available with CMAF ingest, and is enabled by default. To adjust settings, select the MQCS settings on your CMAF channels. For more information about channel settings, see [Creating a channel in AWS Elemental MediaPackage](#).

How MQCS works

The following is a high-level explanation of how MediaPackage uses quality scores from its inputs and in its outputs. For a more detailed explanation about quality scores and how to set up your quality-aware workflow, see the blog [Improve your viewers' live streaming experience with Media Quality-Aware Resiliency](#).

MQCS with inputs to MediaPackage

MQCS in input streams to MediaPackage builds on [live input redundancy](#) functionality. With input redundancy, MediaPackage receives two input streams to a channel. If the active stream is missing any segments, MediaPackage automatically switches to the other stream. MediaPackage continually monitors the streams and switches between them as needed based on segment availability.

With MQCS, the upstream encoder (MediaLive) calculates a quality score for the stream at the ABR-level, based on the health of the source bitstream or elementary, error recovery, and segment errors. The encoder communicates this score to MediaPackage. When the input streams have identical health (segment availability and latency), MediaPackage determines which stream to use for source content based on which has the highest quality score. MediaPackage can effectively provide dynamic segment replacement as it receives content from the encoder.

When both input streams have equal MQCS scores, you can configure MediaPackage to prefer a specific input by setting the **Preferred input** option when creating or updating your channel. This ensures consistent behavior when quality scores are identical, which is particularly useful for cross-Region architectures where multiple MediaPackage channels need to produce identical outputs.

The inputs are identified as follows:

- **Input 1** corresponds to the first ingest endpoint URL returned in the channel creation response
- **Input 2** corresponds to the second ingest endpoint URL returned in the channel creation response

You can also use MQCS to aid in [Cross-Region failover](#) when you enable MediaPackage to include quality scores in responses to requests from downstream systems, such as CDNs (CloudFront). The next section describes how MQCS works with outputs from MediaPackage.

MQCS with outputs from MediaPackage

As MediaPackage receives streams from the encoder, it re-calculates the quality score for each segment and track type. It also calculates an average quality score for all segments in the media sequence. MediaPackage communicates this score through CMSD headers in its responses to downstream systems, such as CDNs (CloudFront) and other quality-monitoring systems.

Downstream systems can use the quality scores from MediaPackage to determine from which origin to serve content, failing over between origins to provide the highest quality viewing experience. CloudFront uses the quality scores as an input to its media quality-aware resiliency functionality (MQAR). For information about MQAR, see [Media quality-aware resiliency](#) in the *Amazon CloudFront Developer Guide*.

Requirements for using MQCS

Your workflow must meet these requirements to use MQCS:

- To enable CDN failover based on quality scores, you must have two identical channels in MediaPackage, with identical origin endpoints.
- Your CDN must support common media server data (CMSD) HTTP headers. If you're using CloudFront, see [Use real-time logs](#) in the *Amazon CloudFront Developer Guide* for more information about CMSD and CMCD headers in logs.
- For resiliency against Region failures, you must set up cross-Region failover. For details and instructions, see the blog [Build a resilient cross-Region live streaming architecture in AWS](#).

Passing through metadata from AWS Elemental MediaPackage

AWS Elemental MediaPackage automatically passes through ID3 metadata from a channel's input to the channel's output stream. You don't need to adjust your endpoint's configuration to enable metadata passthrough.

ID3 metadata considerations

Timed ID3 metadata is a general-purpose mechanism that adds synchronized metadata to streams. The metadata is used for a variety of purposes, ranging from interactive applications to audience measurement.

Supported MediaPackage endpoint types

MediaPackage supports ID3 metadata passthrough for the following endpoint types:

- Live TS and CMAF origin endpoints

Metadata carriage

Here is how ID3 is carried as metadata in the following specifications:

- TS - Metadata is carried in the elementary stream. For more information, see [section 2.0](#) of the *Apple Timed Metadata for HTTP Live Streaming* reference.
- CMAF - Metadata is carried in the Event Message box version 1. For more information, see [Carriage of ID3 Timed Metadata in CMAF](#). Event Message boxes include a `scheme_id_uri` field set to `https://aomedia.org/emsg/ID3` and a `value` field set to `0`.

Metadata signaling

DASH manifests include a `<InbandEventStream schemeIdUri="https://aomedia.org/emsg/ID3" value="0"/>` element in AdaptationSets that include tracks with ID3 metadata.

HLS manifests don't have specific metadata signaling.

MediaLive configuration

You can produce ID3 metadata in AWS Elemental MediaLive [MediaPackage output groups](#) either by [passing through ID3 metadata](#), or [inserting ID3 metadata using the schedule](#).

KLVS metadata considerations

KLVS is a data encoding standard for including synchronized metadata in streams. The binary nature of KLVS makes it efficient when the volume of metadata is significant. KLVS can be used for various use cases ranging from aerial surveillance to transmitting sensors data in industry use cases, or for real-time athlete and object tracking in live sports use cases.

Supported MediaPackage endpoint types

MediaPackage supports KLV metadata passthrough for the following endpoint types:

- Live DASH endpoints

Metadata carriage

Metadata is carried in the Event Message box version 1, as described in the [MISB ST 1910.1 specification](#). For synchronous KLV tracks, Event Message boxes include a `scheme_id_uri` field set to `urn:misb:KLV:bin:1910.1` and a `value` field set to `KLVx:01FC`. For asynchronous KLV tracks, the `value` field is set to `KLVx:01BD`. In both cases, `x` is the index of the track in the stream.

Metadata signaling

DASH manifests include a `<InbandEventStream`
`schemeIdUri="urn:misb:KLV:bin:1910.1" value="KLVx:01FC"/>` or
`<InbandEventStream schemeIdUri="urn:misb:KLV:bin:1910.1" value="KLVx:01BD"/>`
`>` element in `AdaptationSets` that include tracks with KLV metadata, depending on the synchronicity nature of the carried track.

MediaLive configuration

You can pass through KLV metadata from your MediaLive channel. For more information, see [klv](#) in the *AWS Elemental MediaLive User Guide*.

MSS in AWS Elemental MediaPackage

AWS Elemental MediaPackage supports Microsoft Smooth Streaming (MSS) for delivering content to legacy devices such as smart TVs and other platforms where MSS is the preferred or required streaming format. This topic provides an overview of MSS and how it works with MediaPackage.

MSS remains important for reaching legacy devices and platforms that don't support newer protocols, even though HLS and DASH are more common in modern streaming environments.

Similar to other adaptive bitrate streaming protocols like HLS and DASH, MSS dynamically adjusts the quality of a video stream based on network conditions and device capabilities. The key difference is that MSS uses a different format and structure for its manifests and segments, and it was developed by Microsoft specifically for their platforms (like Silverlight, Xbox, etc.).

When to use MSS

MSS is particularly valuable in the following scenarios:

- **Legacy device support:** If your audience uses older smart TVs, set-top boxes, or gaming consoles (particularly Xbox 360) that only support MSS, this format is essential for reaching those viewers.
- **Corporate environments:** Many enterprise environments standardized on Microsoft technologies, including Silverlight-based players, and may not have updated their infrastructure to support newer protocols.
- **Multi-protocol strategy:** For maximum audience reach, content providers often implement a multi-protocol strategy where the same content is available in HLS, DASH, and MSS formats, allowing the client to use whichever format it supports best.
- **Regions with older device penetration:** In some geographic regions, older devices with MSS support remain common, making this format important for reaching those markets.

Choosing the right streaming protocol for your audience

To reach the widest possible audience, select the appropriate streaming protocols based on your viewers' devices. This section helps you decide when to use MSS versus other protocols.

Protocol selection guide

Use this guide to determine which protocols you need to implement:

1. **Identify your audience's devices:** Survey your target audience to understand what devices they use to consume your content.
2. **Check device compatibility:** If your audience includes any of these devices, consider implementing MSS:
 - Xbox 360 consoles
 - Legacy smart TVs (especially older Samsung and LG models)
 - Devices with Microsoft Silverlight players
 - Corporate environments with standardized Microsoft infrastructure
3. **Evaluate audience reach vs. implementation cost:** Determine what percentage of your audience requires MSS support and whether the additional implementation effort is justified.
4. **Consider a multi-protocol approach:** For maximum audience reach, implement multiple protocols and use device detection to serve the appropriate format.

Protocol comparison for implementation planning

This comparison helps you understand the key differences between streaming protocols to plan your implementation:

Streaming Protocol Comparison

Feature	MSS	HLS	DASH
Developer	Microsoft	Apple	MPEG
Year introduced	2008	2009	2012
Manifest format	XML	M3U8 (text)	XML (MPD)
Segment format	MP4 (ISM)	TS or fMP4	MP4
DRM support	PlayReady only	FairPlay, AES-128	Multiple (Widevine, PlayReady, etc.)
Device support	Xbox, Silverlight, older smart TVs	iOS, macOS, newer smart TVs	Android, browsers, newer smart TVs
Industry adoption	Limited (legacy)	Widespread	Growing

Common implementation scenarios

Based on your audience needs, consider these common implementation scenarios:

Modern devices only

If your audience exclusively uses modern devices (smartphones, tablets, newer smart TVs, modern browsers):

- **Recommended:** HLS and DASH
- **Not needed:** MSS

Mixed device ecosystem

If your audience uses a mix of modern and legacy devices:

- **Recommended:** HLS, DASH, and MSS

- **Implementation approach:** Use device detection at the CDN level to route viewers to the appropriate format

Corporate/Enterprise environments

If delivering to corporate environments with standardized Microsoft infrastructure:

- **Recommended:** MSS (primary) with HLS/DASH as alternatives
- **Implementation approach:** Optimize for MSS playback while providing alternatives for BYOD scenarios

While HLS and DASH are now the dominant streaming protocols for modern devices, MSS continues to play an important role in comprehensive streaming strategies that aim to reach the widest possible audience.

Planning your MSS implementation

When you plan your MSS implementation in MediaPackage, you need to understand the technical requirements and considerations to ensure successful delivery to legacy devices.

Technical requirements

Make sure your implementation meets these technical requirements:

- **Container type:** MSS requires the ISM container type. You cannot use CMAF or TS containers with MSS manifests.
- **Encryption:** Only CENC encryption with PlayReady DRM is supported for MSS content. Key rotation is not supported.
- **Audio codecs:** MSS typically uses AAC audio with the AACL FourCC code.
- **Video codecs:** MSS typically uses H.264 video with the H264 FourCC code.

Implementation options

When you configure your MSS endpoints, consider these options:

- **Manifest layouts:** Choose between:
 - **Full layout:** Better compatibility with older players but larger manifest size
 - **Compact layout:** Uses the FragmentRepeat field to significantly reduce manifest size for streams with consistent segment durations

- **Subtitles:** If you need subtitles, use TTML format as MSS does not support WebVTT.

Implementation limitations

Be aware of these limitations when you plan your MSS implementation:

- **SCTE-35:** MSS endpoints do not support SCTE-35 message passthrough or ad marker insertion. If you need ad insertion, consider server-side ad insertion before packaging.
- **DRM options:** Only PlayReady DRM is supported. Key rotation is not supported for MSS content. If you need multi-DRM support, you'll need to implement additional protocols alongside MSS.
- **Subtitle formats:** Limited to TTML subtitles only.

Implementation checklist

Use this checklist when you plan your MSS implementation:

1. Verify your source content is compatible with MSS requirements (codecs, container)
2. Decide on manifest layout based on your audience's devices (Full for maximum compatibility, Compact for efficiency)
3. Understand that LookaheadCount is fixed at 2 and plan your segment duration accordingly for desired latency
4. If encryption is needed, set up a SPEKE key provider for PlayReady DRM
5. Configure your CDN with appropriate cache settings and MIME types for MSS content
6. Test playback on representative devices from your target audience

For step-by-step instructions on creating MSS endpoints, see [Create an MSS manifest](#).

MSS use cases

Consider using MSS endpoints in the following scenarios:

- Delivering content to legacy smart TVs and devices that only support MSS
- Supporting Microsoft platforms where MSS is the preferred streaming format
- Creating multi-protocol streaming workflows that need to include MSS alongside other formats
- Broadcasting to regions where older devices with MSS support are still common

- Supporting Xbox 360 and other legacy Microsoft gaming consoles
- Providing content to corporate environments that standardized on Microsoft Silverlight-based players

Here's a real-world example of MSS implementation:

Example Broadcasting sports events to multiple device types

A major sports broadcaster needed to stream live events to viewers using a wide range of devices, from modern smartphones to older smart TVs and gaming consoles. They implemented a multi-protocol strategy using MediaPackage to create:

- HLS endpoints for iOS devices and newer smart TVs
- DASH endpoints for Android devices and modern browsers
- MSS endpoints for Xbox 360 consoles and older Samsung smart TVs

By including MSS in their delivery strategy, they were able to reach an additional 15% of their audience who were using legacy devices that didn't support newer protocols. The broadcaster configured their CDN to route viewers to the appropriate format based on device detection, ensuring optimal playback for all viewers regardless of their device.

For information about testing MSS playback on compatible devices, see [Testing MSS playback in AWS Elemental MediaPackage](#).

The following sections provide more information about using MSS with MediaPackage.

Topics

- [MSS manifest structure in AWS Elemental MediaPackage](#)
- [MSS encryption and DRM in AWS Elemental MediaPackage](#)
- [Testing MSS playback in AWS Elemental MediaPackage](#)
- [Troubleshooting MSS endpoints in AWS Elemental MediaPackage](#)
- [CDN configuration for MSS in AWS Elemental MediaPackage](#)

MSS manifest structure in AWS Elemental MediaPackage

This topic explains the structure and components of Microsoft Smooth Streaming (MSS) manifests in AWS Elemental MediaPackage. Understanding the manifest structure helps you troubleshoot

playback issues and optimize your MSS streaming workflow. For step-by-step instructions on creating MSS endpoints, see [Create an MSS manifest](#). For information about encrypting MSS content, see [MSS encryption and DRM in AWS Elemental MediaPackage](#).

An MSS manifest is an XML document that describes the available media streams, their qualities, and how to access the media fragments. The main components of an MSS manifest include:

- **SmoothStreamingMedia:** The root element containing metadata about the presentation, including version, timescale, and whether the content is live.
- **Protection:** Contains DRM information when content is encrypted.
- **StreamIndex:** Defines a media stream (video, audio, or text) and its properties.
- **QualityLevel:** Defines the available bitrates and encoding parameters for a stream.
- **c:** Represents a fragment (segment) with attributes for duration (d), timestamp (t), and repeat count (r).

MSS manifests use the MIME type text/xml and are accessed using URLs that end with .ism/Manifest rather than .html.

MSS manifest examples

Here's an example of an MSS manifest with the Full layout:

```
<?xml version="1.0" encoding="utf-8"?>
<SmoothStreamingMedia MajorVersion="2" MinorVersion="2" TimeScale="10000000"
    CanSeek="TRUE" CanPause="TRUE" IsLive="TRUE"
    LookaheadCount="2" DVRWindowLength="600000000" Duration="0">
  <Protection>
    <ProtectionHeader SystemID="9a04f079-9840-4286-ab92-e65be0885f95">
      TG9yZW0gaXBzdW0=
    </ProtectionHeader>
  </Protection>
  <StreamIndex Type="video" Name="video" Subtype="" Chunks="30" TimeScale="10000000"
    Url="QualityLevels({bitrate})/Fragments(v={start time})"
    QualityLevels="2">
    <QualityLevel Index="0" Bitrate="8000000"

    CodecPrivateData="000000001274D401FA9180A00B7602200000300020000030079CD8001E848003D09DEF701F080
      FourCC="H264" MaxWidth="1280" MaxHeight="720"/>
    <QualityLevel Index="1" Bitrate="307200"
```

```

CodecPrivateData="000000001274D401FA9180A00B7602200000300020000030079CD8001E848003D09DEF701F080
    FourCC="H264" MaxWidth="720" MaxHeight="480"/>
    <c d="20000000" t="3414723000000"/>
    <c d="20000000"/>
    <c d="20000000"/>
    <c d="20000000"/>
</StreamIndex>
<StreamIndex Type="audio" Name="eng_1" Language="eng" Subtype="" Chunks="30"
    TimeScale="10000000" Url="QualityLevels({bitrate})/Fragments(a_9_0={start
time})">
    <QualityLevel Index="0" Bitrate="129941" CodecPrivateData="1190" FourCC="AACL"
        AudioTag="255" Channels="2" SamplingRate="48000" BitsPerSample="16"
        PacketSize="4"/>
    <c d="20053333" t="3414723040000"/>
    <c d="20053333"/>
    <c d="20053334"/>
    <c d="19840000"/>
</StreamIndex>
</SmoothStreamingMedia>

```

When using the Compact manifest layout, MediaPackage uses the `FragmentRepeat (r)` attribute to indicate repeated segments with the same duration, significantly reducing manifest size. Here's an example of a compact manifest:

```

<SmoothStreamingMedia MajorVersion="2" MinorVersion="2" TimeScale="10000000"
    Duration="6000" IsLive="FALSE">
    <StreamIndex Type="video" Timescale="10000000" Name="video" Chunks="1"
        QualityLevels="1" Url="http://example.com/video/{bitrate}/{start time}">
        <QualityLevel Index="0" Bitrate="5000000" MaxWidth="1920" MaxHeight="1080"

CodecPrivateData="000000001674D401F965405A03C59722C4840000003004000000CA3C60CA80000000168EE3C80
        FourCC="H264" NALUnitLengthField="4"/>
        <c d="2000" t="0" r="3"/>
    </StreamIndex>
    <StreamIndex Type="audio" Timescale="10000000" Name="audio" Chunks="1"
        QualityLevels="1" Url="http://example.com/audio/{bitrate}/{start time}">
        <QualityLevel Index="0" Bitrate="128000" CodecPrivateData="1190"
            SamplingRate="44100" Channels="2" BitsPerSample="16"
            PacketSize="4" AudioTag="255" FourCC="AACL"/>
        <c d="2000" t="0" r="1"/>
    </StreamIndex>
</SmoothStreamingMedia>

```

```
</StreamIndex>  
</SmoothStreamingMedia>
```

Notice that in the compact format, the `r="3"` attribute indicates that the segment with duration 2000 repeats 3 times, which is more efficient than listing each segment separately.

For more information about configuring manifest layouts when creating MSS endpoints, see [Create an MSS manifest](#). To understand how these manifest formats affect playback on legacy devices, see [Testing MSS playback in AWS Elemental MediaPackage](#).

Key manifest attributes

Key attributes in the MSS manifest that you should understand:

TimeScale

Defines the timescale used for timestamps in the manifest. In the example above, 10000000 means that time values are in units of 1/10,000,000 seconds (0.1 microseconds).

IsLive

Indicates whether the content is live (TRUE) or video-on-demand (FALSE).

LookaheadCount

The number of fragments that must be available before the current fragment is made available in the manifest.

DVRWindowLength

The duration of the DVR window in TimeScale units. In the example above, 600000000 units at a timescale of 10000000 equals 60 seconds.

FourCC

Four-character code that identifies the codec used for the stream. Common values are "H264" for video and "AAC" for audio.

For information about how these attributes affect playback behavior, see [Troubleshooting MSS endpoints in AWS Elemental MediaPackage](#). To learn how to configure these attributes when creating MSS endpoints, see [Create an MSS manifest](#).

MSS segment structure

MSS segments are based on the ISO Base Media File Format (ISOBMFF). Unlike CMAF segments used for DASH and HLS, MSS requires ISM segments which include specific boxes for MSS playback:

- **Tfxd UUID Box:** Encapsulates the absolute timestamp and duration of a fragment in a live presentation.
- **Tfrf UUID Box:** Encapsulates the absolute timestamp and duration for one or more subsequent fragments of the same track in a live presentation (used for lookahead).
- **Senc UUID Box:** Contains the sample-specific encryption data, including the initialization vectors needed for decryption when content is protected.

MSS segments don't have a sequence number like HLS segments. Instead, the player determines the next segment by taking the sum of the previous segment's presentation timestamp and duration.

If you encounter issues with MSS segment playback or availability, see [Troubleshooting MSS endpoints in AWS Elemental MediaPackage](#) for common problems and solutions.

For information about creating MSS endpoints, see [Create an MSS manifest](#).

MSS encryption and DRM in AWS Elemental MediaPackage

This topic explains how to protect your Microsoft Smooth Streaming (MSS) content using encryption and digital rights management (DRM) in AWS Elemental MediaPackage. For information about MSS manifest structure and how encryption affects it, see [MSS manifest structure in AWS Elemental MediaPackage](#). For general information about MSS in MediaPackage, see [MSS in AWS Elemental MediaPackage](#).

MSS encryption in MediaPackage supports PlayReady DRM only. Other DRM systems are not supported for MSS content.

Understanding MSS encryption requirements

When encrypting MSS content in MediaPackage, note the following requirements and limitations:

- **DRM system:** Only PlayReady DRM is supported for MSS content
- **Encryption method:** The encryption method must be CENC (Common Encryption)

- **PlayReady header:** The PlayReady header is inserted at the top level of the manifest
- **Key rotation and constant IV:** Key rotation and constant IV (Initialization Vector) are not supported for MSS content
- **CPIX requirements:** Your key server must provide a `smoothProtectionHeaderData` element in the CPIX response so that MediaPackage can insert it in the manifest. For more information about CPIX, see the [CPIX documentation](#).
- **Shared DRM presets:** Video and audio presets must be set to SHARED because MSS uses the same DRM information across the manifest for video, audio, and subtitle segments

For information about how these encryption options affect your CDN configuration, see [CDN configuration for MSS in AWS Elemental MediaPackage](#). For details on planning your MSS implementation with encryption, see [Planning your MSS implementation](#).

Encrypting your MSS content with PlayReady

To encrypt your MSS content:

1. When creating or editing your endpoint, select **Encrypt content** in the Encryption section.
2. For **Encryption method**, select **CENC**.
3. For **DRM systems**, select **PlayReady**.
4. Ensure that video and audio presets are set to **SHARED** because MSS uses the same DRM information across all segments.
5. Complete the remaining encryption fields as described in [Encryption fields](#).

PlayReady DRM for MSS

PlayReady is Microsoft's DRM technology and is the only supported DRM system for MSS content. When using PlayReady with MSS:

- The PlayReady header is included in the MSS manifest in the `Protection` element
- The PlayReady SystemID is 9a04f079-9840-4286-ab92-e65be0885f95
- The encryption keys are provided by your SPEKE key provider

For more information about configuring SPEKE with MediaPackage, see [Content encryption and DRM in AWS Elemental MediaPackage](#).

Key server requirements

When setting up a key server for MSS content, ensure that:

- The key server has the correct headers that match what is present in the PSSH box
- The UUIDs are properly configured
- Request information is in SOAP/HTTP format. MSS uses the same information consistently across all components - the manifest, key service, audio streams, and video streams all share the same request format and authentication details

For more information about requirements for using PlayReady with MSS, see [Microsoft PlayReady documentation](#).

Unlike some other streaming protocols that may use different authentication or request formats for different components, MSS maintains consistency across all elements. This means that once you configure the SOAP/HTTP request format for your key server, the same format and authentication approach will be used whether the player is requesting the manifest, contacting the key service, or accessing audio and video streams.

MSS encryption limitations

MSS encryption has the following limitations:

- Key rotation is not supported
- Constant IV (Initialization Vector) is not supported

Verifying your encrypted MSS streams

To test your encrypted MSS content:

1. Use the Castlabs Demo Player at <https://demo.castlabs.com/> for testing PlayReady encrypted MSS content
2. Enter your MSS endpoint URL
3. If your content is encrypted with PlayReady DRM, verify that the player can decrypt and play the content. Use the Castlabs player specifically for encrypted content. Other players might not work reliably with PlayReady encrypted MSS streams.

For production environments, ensure your players and devices are properly configured to handle PlayReady-protected content.

For comprehensive testing procedures and compatible players for MSS content, see [Testing MSS playback in AWS Elemental MediaPackage](#).

Testing MSS playback in AWS Elemental MediaPackage

After creating your MSS endpoint in AWS Elemental MediaPackage, you need to test playback to ensure your content is properly delivered to viewers. This topic shows you how to test MSS playback using compatible players and tools. For information about creating MSS endpoints, see [Create an MSS manifest](#). For information about planning your MSS implementation, see [Planning your MSS implementation](#).

MSS compatible players

For testing MSS playback, we recommend using the following players:

- **Dash.js Reference Player** (recommended for unencrypted content): <https://reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html>

Example URL format: `https://reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html?mpd=https://playready.directtaps.net/smoothstreaming/SSWSS720H264/SuperSpeedway_720.ism/Manifest`

- **Castlabs Demo Player** (recommended for encrypted content): <https://demo.castlabs.com/>

Use this player for PlayReady encrypted MSS content. The dash.js player might not work reliably with encrypted MSS streams.

For detailed information about testing encrypted MSS content, see the [MSS DRM Castlabs documentation](#).

MSS is also supported by the following players and platforms:

- Microsoft Silverlight-based players
- Xbox 360 and Xbox One
- Windows Media Player with appropriate extensions
- Certain smart TV models (especially older Samsung and LG models)

- Some set-top boxes that specifically support MSS

When you select a player for testing MSS content, consider the following:

- **Dash.js Reference Player:** Good for general MSS testing and provides diagnostic information about manifest parsing and segment loading.
- **Castlabs Demo Player:** Best for encrypted content testing as it handles PlayReady DRM more reliably than other web-based players.
- **Xbox consoles:** Important for testing if your target audience includes gaming console users. Xbox 360 in particular relies heavily on MSS.
- **Legacy smart TVs:** If targeting older smart TV models, test on actual devices when possible, as their MSS implementations may have specific quirks or limitations.

Testing procedure

Follow these steps to test your MSS endpoint:

1. Get your MSS endpoint URL from the MediaPackage console. The URL will end with .ism.
2. Open the Dash.js Reference Player or Castlabs Demo Player (for encrypted content).
3. Enter your MSS endpoint URL in the player.
4. Verify that playback starts and the content plays smoothly.
5. If your content is encrypted with PlayReady DRM, verify that the player can decrypt and play the content.

For a comprehensive test of your MSS implementation, consider these additional testing steps:

- **Network condition testing:** Test playback under various network conditions (bandwidth throttling, packet loss, latency) to verify adaptive bitrate switching works correctly.
- **Device compatibility matrix:** Create a compatibility matrix of target devices and test on representative samples from each category.
- **Long-duration testing:** For live streams, conduct extended playback tests (4+ hours) to verify there are no issues with manifest updates or segment availability over time.
- **CDN edge case testing:** Test from different geographic locations to ensure your CDN configuration works properly across all regions.

Example Real-world testing scenario: Live sports broadcast

A media company preparing for a major sports tournament implemented the following testing strategy for their MSS streams:

1. Pre-event testing with simulated content to verify endpoint configuration
2. Dress rehearsal with actual live content one week before the event
3. Testing on 5 different device types representing their audience demographics
4. Network simulation tests with bandwidth fluctuations to verify adaptive bitrate behavior
5. Monitoring setup to track manifest requests, segment delivery, and error rates during the event

This comprehensive approach helped them identify and resolve several issues before the live event, including a CDN configuration problem that was causing intermittent 404 errors for MSS segments.

For information about monitoring your MSS streams during testing and production, see [Monitoring MSS streams](#). For information about testing encrypted MSS content, see [Verifying your encrypted MSS streams](#).

CDN configuration for testing

For production environments, ensure your CDN is properly configured to handle MSS content:

- Set appropriate cache control headers for both manifests (.ism) and segments
- Configure proper MIME types: application/vnd.ms-sstr+xml for manifests and video/mp4 for segments
- If using encryption, ensure your CDN is configured to handle PlayReady DRM

For detailed CDN configuration instructions specific to MSS, see [CDN configuration for MSS in AWS Elemental MediaPackage](#).

If you encounter issues during testing, see [Troubleshooting MSS endpoints in AWS Elemental MediaPackage](#) for common problems and solutions.

Troubleshooting MSS endpoints in AWS Elemental MediaPackage

This topic helps you identify and resolve common issues with Microsoft Smooth Streaming (MSS) endpoints in AWS Elemental MediaPackage.

Resolving common MSS playback issues

If you encounter issues with your MSS endpoints, consider the following common problems and solutions:

404 errors when requesting segments

This could be due to the `LookaheadCount` setting. MSS holds back segments at the live edge until 2 future segments are available (this value is fixed and not configurable).

Solution: Ensure your player is requesting segments within the available window, accounting for the fixed 2-segment lookahead buffer.

Playback issues on legacy devices

Some older devices may have limitations with certain MSS features.

Solution: Try using the **Full** manifest layout instead of **Compact**, as it's more widely supported by legacy players.

DRM issues

MSS only supports PlayReady DRM.

Solution: Ensure your SPEKE key provider is correctly configured to provide PlayReady keys. For more information, see [Content encryption and DRM in AWS Elemental MediaPackage](#).

Manifest not loading

The MSS manifest may not load if the endpoint is not properly configured.

Solution: Verify that you've selected the ISM container type when creating your endpoint. MSS manifests require the ISM container type.

Playback stuttering or buffering

This could be due to network issues or segment availability.

Solution: Check your network connection and ensure your CDN is properly configured for MSS content. Also, verify that your segment duration is appropriate for your content and network conditions.

For a better understanding of MSS manifest structure to help diagnose issues, see [MSS manifest structure in AWS Elemental MediaPackage](#).

Monitoring MSS streams

Effective monitoring of your MSS streams helps you ensure reliable delivery and quickly identify issues before they impact your viewers. This section provides detailed guidance on setting up comprehensive monitoring for MSS endpoints.

Key metrics to monitor

When monitoring MSS streams, pay attention to these key metrics:

Request count

Track the number of requests for manifests and segments to understand your audience size and viewing patterns.

Normal pattern: MSS clients typically request manifests less frequently than HLS clients, with a ratio of approximately 1 manifest request to 10-20 segment requests.

Warning signs: Sudden drops in request count may indicate playback issues or CDN problems.

HTTP response codes

Monitor the distribution of HTTP status codes to identify potential issues.

Normal pattern: Primarily 200 OK responses, with some 412 errors near the live edge due to the lookahead buffer (these are expected).

Warning signs: High rates of 5xx errors, unexpected 403 errors, or 404 errors for established content.

Latency

Track the time it takes to serve manifest and segment requests.

Normal pattern: Manifest requests typically have higher latency than segment requests due to dynamic generation.

Warning signs: Increasing latency trends or spikes above 500ms for segment delivery.

Egress bytes

Monitor the volume of data being delivered to understand bandwidth usage and costs.

Normal pattern: Consistent patterns that align with your audience's viewing habits.

Warning signs: Unexpected spikes or drops that don't correlate with audience size changes.

Setting up a CloudWatch dashboard for MSS

Create a dedicated Amazon CloudWatch dashboard to monitor your MSS endpoints with these recommended widgets:

1. Request metrics:

```
{
  "metrics": [
    [ "AWS/MediaPackage", "EgressRequestCount", "Channel", "YourChannelName",
      "Origin", "YourOriginEndpointName", { "stat": "Sum", "period": 60 } ]
  ],
  "view": "timeSeries",
  "stacked": false,
  "region": "us-west-2",
  "title": "MSS Endpoint Requests",
  "period": 300
}
```

2. HTTP status code distribution:

```
{
  "metrics": [
    [ "AWS/MediaPackage", "EgressRequestCount", "Channel", "YourChannelName",
      "Origin", "YourOriginEndpointName", "StatusCodeRange", "4xx", { "stat": "Sum",
      "period": 60 } ],
    [ "AWS/MediaPackage", "EgressRequestCount", "Channel", "YourChannelName",
      "Origin", "YourOriginEndpointName", "StatusCodeRange", "5xx", { "stat": "Sum",
      "period": 60 } ],
    [ "AWS/MediaPackage", "EgressRequestCount", "Channel", "YourChannelName",
      "Origin", "YourOriginEndpointName", { "stat": "Sum", "period": 60 } ]
  ],
  "view": "timeSeries",
  "stacked": true,
  "region": "us-west-2",
  "title": "MSS HTTP Status Codes",
}
```

```
"period": 300
}
```

3. Latency tracking:

```
{
  "metrics": [
    [ "AWS/MediaPackage", "EgressResponseTime", "Channel", "YourChannelName",
      "Origin", "YourOriginEndpointName", { "stat": "Average", "period": 60 } ],
    [ "AWS/MediaPackage", "EgressResponseTime", "Channel", "YourChannelName",
      "Origin", "YourOriginEndpointName", { "stat": "p90", "period": 60 } ],
    [ "AWS/MediaPackage", "EgressResponseTime", "Channel", "YourChannelName",
      "Origin", "YourOriginEndpointName", { "stat": "p99", "period": 60 } ]
  ],
  "view": "timeSeries",
  "stacked": false,
  "region": "us-west-2",
  "title": "MSS Endpoint Latency",
  "period": 300
}
```

4. Egress tracking:

```
{
  "metrics": [
    [ "AWS/MediaPackage", "EgressBytes", "Channel", "YourChannelName", "Origin",
      "YourOriginEndpointName", { "stat": "Sum", "period": 60 } ]
  ],
  "view": "timeSeries",
  "stacked": false,
  "region": "us-west-2",
  "title": "MSS Egress Bytes",
  "period": 300
}
```

Replace `YourChannelName` and `YourOriginEndpointName` with your actual channel and endpoint names, and adjust the region as needed.

Recommended CloudWatch alarms

Set up these Amazon CloudWatch alarms to proactively detect issues with your MSS streams:

High error rate alarm

Triggers when the percentage of 4xx or 5xx errors exceeds a threshold.

```
aws cloudwatch put-metric-alarm \  
  --alarm-name MSS_HighErrorRate \  
  --alarm-description "Alarm when MSS endpoint error rate exceeds 5%" \  
  --metrics '[  
    {  
      "Id": "e1",  
      "Expression": "(m2+m3)/m1*100",  
      "Label": "Error Rate"  
    },  
    {  
      "Id": "m1",  
      "MetricStat": {  
        "Metric": {  
          "Namespace": "AWS/MediaPackage",  
          "MetricName": "EgressRequestCount",  
          "Dimensions": [  
            { "Name": "Channel", "Value": "YourChannelName" },  
            { "Name": "Origin", "Value": "YourOriginEndpointName" }  
          ]  
        },  
        "Period": 300,  
        "Stat": "Sum"  
      },  
      "ReturnData": false  
    },  
    {  
      "Id": "m2",  
      "MetricStat": {  
        "Metric": {  
          "Namespace": "AWS/MediaPackage",  
          "MetricName": "EgressRequestCount",  
          "Dimensions": [  
            { "Name": "Channel", "Value": "YourChannelName" },  
            { "Name": "Origin", "Value": "YourOriginEndpointName" },  
            { "Name": "StatusCodeRange", "Value": "4xx" }  
          ]  
        },  
        "Period": 300,  
        "Stat": "Sum"  
      },  
      "ReturnData": false  
    },  
    {  
      "Id": "m3",  
      "MetricStat": {  
        "Metric": {  
          "Namespace": "AWS/MediaPackage",  
          "MetricName": "EgressRequestCount",  
          "Dimensions": [  
            { "Name": "Channel", "Value": "YourChannelName" },  
            { "Name": "Origin", "Value": "YourOriginEndpointName" },  
            { "Name": "StatusCodeRange", "Value": "5xx" }  
          ]  
        },  
        "Period": 300,  
        "Stat": "Sum"  
      },  
      "ReturnData": false  
    }  
  ]
```

```

    ]
  },
  "Period": 300,
  "Stat": "Sum"
},
"ReturnData": false
},
{
  "Id": "m3",
  "MetricStat": {
    "Metric": {
      "Namespace": "AWS/MediaPackage",
      "MetricName": "EgressRequestCount",
      "Dimensions": [
        { "Name": "Channel", "Value": "YourChannelName" },
        { "Name": "Origin", "Value": "YourOriginEndpointName" },
        { "Name": "StatusCodeRange", "Value": "5xx" }
      ]
    },
    "Period": 300,
    "Stat": "Sum"
  },
  "ReturnData": false
}
]' \
--threshold 5 \
--comparison-operator GreaterThanThreshold \
--evaluation-periods 2 \
--alarm-actions [your-sns-topic-arn]

```

Latency spike alarm

Triggers when the average latency exceeds a threshold.

```

aws cloudwatch put-metric-alarm \
  --alarm-name MSS_HighLatency \
  --alarm-description "Alarm when MSS endpoint latency exceeds 500ms" \
  --metric-name EgressResponseTime \
  --namespace AWS/MediaPackage \
  --statistic Average \
  --period 300 \
  --threshold 500 \

```

```
--comparison-operator GreaterThanThreshold \  
--dimensions Name=Channel,Value=YourChannelName  
Name=Origin,Value=YourOriginEndpointName \  
--evaluation-periods 3 \  
--alarm-actions [your-sns-topic-arn]
```

Request drop alarm

Triggers when request count drops significantly, which could indicate delivery issues.

```
aws cloudwatch put-metric-alarm \  
--alarm-name MSS_RequestDrop \  
--alarm-description "Alarm when MSS requests drop by more than 50%" \  
--metric-name EgressRequestCount \  
--namespace AWS/MediaPackage \  
--statistic Sum \  
--period 300 \  
--threshold 0 \  
--comparison-operator LessThanThreshold \  
--dimensions Name=Channel,Value=YourChannelName  
Name=Origin,Value=YourOriginEndpointName \  
--evaluation-periods 2 \  
--alarm-actions [your-sns-topic-arn] \  
--treat-missing-data breaching
```

Interpreting monitoring data

Understanding common patterns in your monitoring data can help you quickly identify and resolve issues:

Pattern: Spike in 404 errors

Possible causes:

- Segment duration too short relative to the fixed 2-fragment lookahead buffer
- Input stream interruption causing gaps in available segments
- CDN cache configuration issues

Recommended action: Check input stream health, ensure segment duration works well with the fixed 2-fragment lookahead buffer, and review CDN cache settings.

Pattern: Increasing latency trend

Possible causes:

- Growing audience size exceeding capacity
- CDN origin shield not properly configured
- Network congestion between CDN and origin

Recommended action: Review your CDN configuration, consider implementing or optimizing origin shield settings, and check for network bottlenecks.

Pattern: Cyclical spikes in request count

Possible causes:

- Normal audience behavior patterns (e.g., primetime viewing)
- CDN cache expiration causing request floods

Recommended action: If these align with expected audience patterns, this is normal. If not, review your CDN TTL settings to ensure they're appropriate for your content type.

For comprehensive information about MediaPackage monitoring capabilities, see [Logging and monitoring in MediaPackage](#).

Common MSS error codes

Understanding the specific HTTP error codes returned by MSS endpoints helps you diagnose and resolve issues more effectively:

404 Not Found

Occurs when the player requests a segment that cannot be found. This typically happens when the requested segment is not available or the URL is incorrect.

Common causes:

- Player requesting segments beyond the available window
- LookaheadCount settings causing segments to be held back
- Input stream interruptions creating gaps in available content

412 Precondition Failed

Occurs when lookahead fragments are not available for the requested segment. This indicates that MediaPackage does not have the lookahead fragments internally available.

Common causes:

- Player requesting segments too close to the live edge
- Input stream issues preventing lookahead fragments from being generated
- Network delays affecting fragment availability

403 Forbidden

Occurs when access to the requested resource is denied.

Common causes:

- CDN authentication or authorization issues
- Incorrect endpoint permissions or access policies
- Geographic restrictions or IP blocking

5xx Server Errors

Indicates server-side issues with the MSS endpoint or underlying infrastructure.

Common causes:

- Service capacity issues or overload
- Backend service failures or timeouts
- Configuration errors in the endpoint setup

Using diagnostic tools to identify MSS issues

The following tools can help you troubleshoot MSS streaming issues:

- Browser developer tools to inspect network requests and responses
- Dash.js or Bitmovin player to verify playback and DRM functionality
- Amazon CloudWatch to monitor endpoint metrics and logs

If you continue to experience issues with your MSS endpoints after trying these solutions, contact AWS Support for assistance.

CDN configuration for MSS in AWS Elemental MediaPackage

This topic explains how to configure your Content Delivery Network (CDN) to properly handle Microsoft Smooth Streaming (MSS) content from AWS Elemental MediaPackage.

Optimizing cache settings for MSS delivery

Configure appropriate cache control headers for MSS content:

Live manifests

Set a short TTL (Time To Live) for live manifests, typically 5-10 seconds, to ensure viewers receive updated content.

Example: `Cache-Control: max-age=5`

VOD manifests

For video-on-demand content, you can set longer TTLs for manifests, typically 60 seconds or more.

Example: `Cache-Control: max-age=60`

Segments

Set longer TTLs for segments since they don't change once created.

Example: `Cache-Control: max-age=86400` (24 hours)

MIME types for MSS content

Configure your CDN to serve MSS content with the correct MIME types:

MSS manifests

MIME type: `text/xml`

File pattern: `*.ism/Manifest`

MSS segments

MIME type: `video/mp4`

File pattern: *Fragments*

Configuring your CDN for PlayReady DRM

If your MSS content is encrypted with PlayReady DRM:

- Ensure your CDN passes through the PlayReady header in the manifest without modification
- Configure your CDN to handle the proper CORS (Cross-Origin Resource Sharing) headers if your players require them
- Verify that your CDN doesn't modify the encrypted segments in any way

For detailed information about MSS encryption and PlayReady DRM, see [MSS encryption and DRM in AWS Elemental MediaPackage](#).

Setting up Amazon CloudFront for MSS delivery

Amazon CloudFront (CloudFront) works well with MSS content from MediaPackage. To configure CloudFront for MSS:

1. Create a new CloudFront distribution or use an existing one
2. Set the origin to your MediaPackage endpoint domain
3. Configure cache behaviors with the appropriate TTLs for manifests and segments
4. If using DRM, ensure that CloudFront is configured to forward the necessary headers

Here's a real-world example of CloudFront configuration for MSS content:

Example CloudFront configuration for a global MSS deployment

A global media company delivering MSS content to legacy devices across multiple regions implemented the following CloudFront configuration:

- **Origin settings:**
 - Origin domain: their MediaPackage endpoint domain
 - Origin protocol policy: Match viewer
 - Origin keep-alive timeout: 5 seconds
- **Cache behavior for manifests (.ism/Manifest):**

- Path pattern: *.ism/Manifest
- TTL settings: Minimum 0, Default 5, Maximum 10 (seconds)
- Compress objects automatically: Yes
- **Cache behavior for segments:**
 - Path pattern: *Fragments*
 - TTL settings: Minimum 3600, Default 86400, Maximum 604800 (seconds)
 - Compress objects automatically: No (to avoid modifying encrypted segments)

This configuration ensured that manifests were refreshed frequently while segments were cached for longer periods, optimizing both content freshness and CDN cost efficiency. The company also implemented regional price classes in CloudFront to balance performance and cost based on their audience distribution.

For Microsoft Smooth Streaming endpoints, you should create a cache behavior with the path pattern `out/v1/your-endpoint-id/index.ism/*` to properly route manifest requests.

For more information about using CloudFront with MediaPackage, see [Working with AWS Elemental MediaPackage and CDNs](#).

AWS Elemental MediaPackage rendition groups reference

Rendition groups are used in TS and CMAF outputs from MediaPackage. A rendition group collects all subtitle or audio tracks and makes them available for all video renditions in the stream. When you enable rendition groups, MediaPackage pulls together all audio variants (such as different languages or codecs) and groups them for use with any video rendition. MediaPackage automatically puts subtitles into a rendition group.

Audio and subtitles tracks are required to be in their own rendition groups for CMAF outputs.

The following sections further describe when you can use rendition groups.

Note

DASH doesn't use rendition groups. This is because all audio, video, and subtitle or caption tracks are presented individually to the player, and the player determines which are used during playback.

When to use rendition groups

Rendition groups are used only in TS and CMAF outputs. Rendition groups are most beneficial when you have multiple languages or multiple audio codecs in your streams. Rendition groups should be used in the following use cases:

- With CMAF outputs, if there are any audio or subtitle tracks

CMAF requires all audio tracks in one rendition group, and all subtitles in another. Audio or subtitles can't be muxed with video tracks.

- One or more video tracks with multiple audio languages or codecs

When rendition groups are enabled, MediaPackage pulls all audio renditions together for shared use between the video tracks. In this way, you don't have to duplicate all the audio options across all the video tracks.

- Multiple audio-only tracks and multiple subtitle tracks

When both the audio tracks and subtitle tracks are in rendition groups, all the audio options can be combined with any subtitle track.

- One audio-only track and multiple subtitle tracks

MediaPackage automatically pulls subtitle tracks into a rendition group so that the audio track can be used with any subtitle. Because there is only one audio and the subtitles are already grouped, you don't need to tell MediaPackage to use rendition groups in this case.

When not to use rendition groups

Rendition groups can't or shouldn't be used in the following use cases:

- Multiple video tracks in the stream, but only one language or codec is used for the audio. If the same audio is used with multiple video tracks, and rendition groups are also used, then your rendition group will have duplicates of the same audio track (one for each video).

Keep the audio and video muxed in the stream, and do not use a rendition group.

- DASH outputs. These protocols do not support rendition groups. Instead, the output stream includes all tracks, and the player determines which to play based on rules from the player side or from the manifest (such as language or bitrate selection).

To limit the tracks available to a player, use the stream selection options from the MediaPackage console or the MediaPackage API.

Reset for AWS Elemental MediaPackage channels and endpoints

There might come a time in your content streaming workflow that you need to reset the content that MediaPackage has ingested or is outputting.

Channel reset uses

For channels (ingest), a reset removes previously ingested content. When you reset the channel, you must also stop the stream from the encoder. Thirty seconds after the channel reset, start the stream from the encoder. With the old content removed from the channel and the ingest stream restarted, you can clear out content that has issues.

Resetting a channel is especially useful when you receive 409 errors from your encoder, which are often caused by encoder reconfigurations that change the numbering logic of CMAF ingest segments. In cross-Region workflows, this logic change can cause incompatibility between new segments and old because segment numbering starts to go backward.

Channel reset is also useful when you switch between input sources with embedded time codes, and input sources without.

Origin endpoint reset uses

For origin endpoints (output), a reset clears out previous content to refresh the output stream. This can help remove content that causes unexpected behaviors, content from a special event, and other situations when you don't want the previous content to be available for viewing.

Resetting an endpoint is especially useful with event-based endpoints. If the specified manifest length reaches beyond the current running time of the content, customers could view content from previous events.

Endpoint reset is also useful when the previous content caused unexpected behaviors, like what could happen if the stream has multiple stream set changes or interruptions in the upstream content.

For steps to reset a channel or endpoint, see the following sections:

- [Resetting channel history](#)
- [Resetting an endpoint](#)

SCTE-35 message options in AWS Elemental MediaPackage

This section describes the options that AWS Elemental MediaPackage offers for configuring how SCTE-35 messages are handled in live TS and CMAF outputs.

SCTE-35 messages accompany video in your source content. These messages signal where MediaPackage should insert ad markers when it packages the content for output. By default, MediaPackage inserts markers for the following message types in the source content:

- Splice insert
- Break
- Provider advertisement
- Distributor advertisement
- Provider placement opportunity
- Distributor placement opportunity
- Provider overlay placement opportunity
- Distributor overlay placement opportunity
- Program
- Chapter
- Unscheduled event
- Alternate content opportunity
- Network
- Provider promo
- Distributor promo
- Provider ad block
- Distributor ad block

When these commands are present, MediaPackage inserts corresponding ad markers in the output manifests:

- For daterange in HLS manifests on TS and CMAF origin endpoints, MediaPackage inserts EXT-X-DATERANGE tags.
- For SCTE-35 enhanced on TS and CMAF origin endpoints, MediaPackage inserts EXT-X-CUE-OUT, EXT-X-CUE-IN, and related CUE tags.
- For DASH manifests on CMAF origin endpoints, MediaPackage inserts EventStream tags to create multiple periods, when you have multi-period manifests enabled.

The following sections describe how you can modify MediaPackage SCTE-35 message handling behavior.

Topics

- [How SCTE-35 messages work in AWS Elemental MediaPackage works](#)
- [SCTE-35 settings in MediaPackage](#)
- [HLS EXT-X-DATERANGE ad markers in AWS Elemental MediaPackage](#)
- [HLS CUE tag ad markers in AWS Elemental MediaPackage](#)
- [DASH ad markers in AWS Elemental MediaPackage](#)

How SCTE-35 messages work in AWS Elemental MediaPackage works

The **Ad markers** and **SCTE filtering** settings work together to determine what MediaPackage does with SCTE-35 messages from the source content.

When there are SCTE-35 messages in the source content, MediaPackage takes the following actions based on the value that you selected in **Ad markers**:

- For **Daterange**, MediaPackage inserts EXT-X-DATERANGE tags to signal ads and program transition events in HLS output manifests.
- For **SCTE-35 enhanced**, MediaPackage inserts EXT-X-CUE-OUT, EXT-X-CUE-IN, and related CUE tags to signal ads and program transition events in HLS output manifests.
- For **XML** or **Binary**, MediaPackage inserts EventStream elements in the DASH manifests, with the selected signaling type, and create new periods when detecting markers defined as Ad markers.

Important note on SCTE-35 data tracks

MediaPackage also signals SCTE-35 markers present in the source that are not ad markers. MediaPackage selects the first available data track from the input content for SCTE-35 signal processing (typically identified as PID 500). For proper handling by MediaPackage, ensure that your SCTE-35 ad signals are included in this first data track.

For DASH manifests, SCTE-35 markers that aren't ad markers don't create a new period. Instead, MediaPackage places these markers in the period that corresponds to the marker timing.

SCTE-35 settings in MediaPackage

You can modify how MediaPackage interacts with SCTE-35 messages from your source content. Configure the following settings on your origin endpoints. For more information, see the following:

- For the MediaPackage console, see [the section called “Creating an origin endpoint”](#).
- For the MediaPackage API, see [CreateOriginEndpoint](#) in the *AWS Elemental MediaPackage Live API Reference*.

Important

To modify how MediaPackage handles SCTE-35 messages, you should be familiar with the SCTE-35 standard. You can view the most recent standards here: [SCTE Standards Catalog](#). You should also be familiar with how SCTE-35 is implemented in your source content.

The SCTE configuration is achieved through settings available both at the segment level and at the manifest level.

Enable SCTE support

This setting is available on TS and CMAF origin endpoints, in the Segment settings parameters group. When enabled, it allows to define the SCTE configuration options in both the Segment settings and Manifest definitions parameters groups.

SCTE filtering

This setting is available on TS and CMAF origin endpoints.

SCTE filtering specifies which SCTE-35 message types MediaPackage uses to create new periods in the output manifest. All SCTE-35 markers are preserved and passed through to the

endpoint manifest, but only the markers that match your specified filter will trigger period boundaries.

If you don't change this setting, MediaPackage treats these message types as ads:

- Splice insert
- Break
- Provider advertisement
- Distributor advertisement
- Provider placement opportunity
- Distributor placement opportunity
- Provider overlay placement opportunity
- Distributor overlay placement opportunity
- Program
- Chapter
- Unscheduled event
- Alternate content opportunity
- Network
- Provider promo
- Distributor promo
- Provider ad block
- Distributor ad block

The default filter includes the first nine values in the preceding list. The remaining values (Chapter, Unscheduled event, Alternate content opportunity, Network, Provider promo, Distributor promo, Provider ad block, and Distributor ad block) require explicit opt-in through the SCTE filtering configuration.

Ad markers

This setting is available for both HLS and DASH manifests, in the SCTE configuration section of the Manifest definitions parameters group.

Ad markers allows you to specify what MediaPackage does when it detects SCTE-35 messages. These are the options for HLS manifests:

- **Daterange** - inserts EXT-X-DATERANGE tags with SCTE-35 data

- **SCTE-35 enhanced** - inserts EXT-X-CUE-OUT, EXT-X-CUE-IN, and related CUE tags

For DASH manifests on CMAF endpoints, these are the options:

- **XML** - inserts EventStream elements with the `urn:scte:scte35:2013:xml` scheme
- **Binary** - inserts EventStream elements with the `urn:scte:scte35:2014:xml+bin` scheme

SCTE in segments

Choose whether to include SCTE-35 messages in the segment files of MediaPackage outputs. This setting works independently of the **SCTE filtering** and **Ad markers** settings. Choose from the following options:

- **None** – SCTE-35 messages are not included in segment output. This is the default setting.
- **All** – All SCTE-35 messages are embedded in the segment data, allowing downstream players and ad insertion systems to access the messages directly from segments without parsing manifest files.
- **Matches filter** – Only SCTE-35 messages from events whose type matches the configured SCTE filtering are embedded in the segment data. This option allows you to limit which SCTE-35 messages are included in segments based on your filter configuration.

When you enable SCTE in segments:

- For TS segments, SCTE-35 messages are embedded directly in the transport stream.
- For CMAF segments, SCTE-35 messages are embedded in the segment data.
- For DASH manifests, when you set SCTE in segments to **All** or **Matches filter**, MediaPackage adds an InbandEventStream element with `schemeIdUri="urn:scte:scte35:2013:bin"` to signal the presence of SCTE-35 messages in the segment data.

Important

Your downstream components or players must be implemented to consume SCTE-35 messages from segments to take advantage of this feature. If your players cannot handle SCTE-35 messages in segments, playback may fail when this setting is enabled.

Custom ad types

This setting is available on TS and CMAF origin endpoints, in the Segment settings parameters group.

Custom ad types allows you to designate additional SCTE-35 event types as advertisements. By default, MediaPackage treats only splice insert and time signal placement opportunity events as ads. Use this setting to expand the set of event types that MediaPackage treats as ads.

Choose from the following values:

- PROGRAM
- CHAPTER
- UNSCHEDULED_EVENT
- ALTERNATE_CONTENT_OPPORTUNITY
- NETWORK

SCTE in manifests

This setting is available for HLS, LL-HLS, and DASH manifests, in the SCTE configuration section of the Manifest definitions parameters group.

SCTE in manifests controls which SCTE-35 events appear in the output manifests. Choose from the following options:

- **All** – All SCTE-35 events appear in the manifest. This is the default setting.
- **Matches filter** – Only SCTE-35 events whose type matches the configured SCTE filtering appear in the manifest. You must have SCTE support enabled with at least one SCTE filter value configured to use this option.

Note

This setting is not available for MSS manifests.

HLS EXT-X-DATERANGE ad markers in AWS Elemental MediaPackage

Daterange ad markers are used to signal ads and program transitions in live HLS manifests. When you enable daterange ad markers on your origin endpoint, MediaPackage inserts EXT-X-DATERANGE tags into the manifest where there are SCTE-35 `time_signal` or `splice_insert` tags present. EXT-X-DATERANGE is used in concert with EXT-X-PROGRAM-DATE-TIME tags.

For information about the EXT-X-DATERANGE and EXT-X-PROGRAM-DATE-TIME tags for HLS, see the [HTTP Live Streaming 2nd Edition Specification](#).

The following table describes how to enable daterange ad markers in MediaPackage.

Console

To enable daterange ad markers when creating or editing an origin endpoint, in the MediaPackage console, under the HLS or LL-HLS manifest settings, **SCTE configuration, Ad markers**, choose **Daterange**.

If you choose daterange, you *must* also enter a **Program date/time interval (sec.)** value of 1 or greater. The program date/time interval is set in the same manifest fields as the ad marker settings.

AWS CLI

To enable daterange ad markers for your origin endpoint, run the following command in the AWS CLI replacing *region* with your own information:

```
aws --endpoint=https://mediapackagev2.region.amazonaws.com mediapackage --  
region region create-origin-endpoint --channel-id test_channel --id hlsmuxed  
--hls-package '{"ProgramDateTimeIntervalSeconds":60,"AdMarkers":{"DATERANGE  
\}'"
```

Important

You must set a ProgramDateTimeIntervalSeconds value that's greater than 0 (zero).

API and SDK

To learn how to enable daterange ad markers for TS and CMAF origin endpoints via the MediaPackage live API or AWS SDK, see the following:

- [MediaPackage Live API reference](#)
- [AWS SDK](#)

Enabling daterange via the MediaPackage API or AWS SDK

To learn how to enable daterange ad markers for TS and CMAF origin endpoints via the MediaPackage live API or AWS SDK, see the following:

- [MediaPackage Live API reference](#)
- [AWS SDK](#)

Example HLS manifest showing SCTE-35 EXT-X-DATERANGE signaling

This example shows a HLS manifest generated by MediaPackage using EXT-X-DATERANGE and EXT-X-PROGRAM-DATE-TIME tags to signal events in the live stream.

Note

The DURATION, PLANNED-DURATION, and END-DATE attributes of the EXT-X-DATERANGE tag are optional. If these attributes aren't present in the SCTE-35 input, or aren't set when you create your origin endpoint via the MediaPackage API, then they are omitted from the generated manifests.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:8
#EXT-X-MEDIA-SEQUENCE:11
#EXT-X-DATERANGE:ID="2415919105",START-DATE="2020-05-03T00:01:00.018Z",PLANNED-
DURATION=29.988,SCTE35-
OUT=0xFC303000000002CDE400FFF00506FE00526C14001A021843554549900000017FC00000292EA80A04ABCD00013
#EXT-X-DATERANGE:ID="2147483649",START-DATE="2020-05-03T00:00:30.030Z",PLANNED-
DURATION=90.006,SCTE35-
CMD=0xFC303000000002CDE400FFF00506FE00293D6C001A021843554549800000017FFF00007B9ABC0A04ABCD00011
#EXT-X-PROGRAM-DATE-TIME:2020-05-03T00:01:08.040Z
#EXTINF:7.560,
index_1_11.ts?m=1588607409
#EXTINF:7.560,
index_1_12.ts?m=1588607409
#EXTINF:6.846,
index_1_13.ts?m=1588607409
#EXT-X-DATERANGE:ID="2415919105",START-DATE="2020-05-03T00:01:00.018Z",END-
DATE="2020-05-03T00:01:30.006Z",DURATION=29.988
#EXTINF:0.714,
index_1_14.ts?m=1588607409
#EXTINF:7.560,
index_1_15.ts?m=1588607409
#EXTINF:7.560,
```

```

index_1_16.ts?m=1588607409
#EXTINF:7.560,
index_1_17.ts?m=1588607409
#EXTINF:6.636,
index_1_18.ts?m=1588607409
#EXT-X-DATERANGE:ID="2147483649",START-DATE="2020-05-03T00:00:30.030Z",END-
DATE="2020-05-03T00:02:00.036Z",DURATION=90.006,SCTE35-
CMD=0xFC304A00000002CDE400FFF00506FE00A4D8280034021843554549800000017FC000000000000A04ABCD00011
#EXT-X-DATERANGE:ID="2147483650",START-DATE="2020-05-03T00:02:00.036Z",PLANNED-
DURATION=90.006,SCTE35-
CMD=0xFC304A00000002CDE400FFF00506FE00A4D8280034021843554549800000017FC000000000000A04ABCD00011
#EXTINF:0.924,
index_1_19.ts?m=1588607409
#EXTINF:7.560,
index_1_20.ts?m=1588607409
#EXT-X-PROGRAM-DATE-TIME:2020-05-03T00:02:08.520Z
#EXTINF:7.560,
index_1_21.ts?m=1588607409
#EXT-X-ENDLIST

```

HLS CUE tag ad markers in AWS Elemental MediaPackage

CUE tag ad markers are used to signal ads and program transitions in live HLS manifests using industry-standard CUE tags. When you enable SCTE-35 enhanced ad markers on your HLS or LLHLS origin endpoint, MediaPackage inserts EXT-X-CUE-OUT, EXT-X-CUE-IN, and related CUE tags into the manifest where there are SCTE-35 `time_signal` or `splice_insert` tags present. This format provides the same SCTE-35 information as DATERANGE tags but in a different manifest format that's compatible with legacy ad insertion systems.

The CUE tags generated by MediaPackage include:

- EXT-X-CUE-OUT - Signals the start of an ad break
- EXT-X-CUE-IN - Signals the end of an ad break
- EXT-X-CUE-OUT-CONT - Continuation of ad break across segments
- EXT-OATCLS-SCTE35 - OATCLS-specific SCTE-35 data
- EXT-X-ASSET - Asset identification tags

For information about CUE tags for HLS, see the [HTTP Live Streaming 2nd Edition Specification](#).

The following table describes how to enable CUE tag ad markers in MediaPackage.

Console

To enable CUE tag ad markers when creating or editing an origin endpoint, in the MediaPackage console, under the HLS or LL-HLS manifest settings, **SCTE configuration, Ad markers**, choose **SCTE-35 enhanced**.

AWS CLI

To enable CUE tag ad markers for your origin endpoint, run the following command in the AWS CLI replacing *region* with your own information:

```
aws --endpoint=https://mediapackagev2.region.amazonaws.com mediapackagev2 --
region region create-origin-endpoint --channel-group-name test_channel_group --
channel-name test_channel --origin-endpoint-name hls_endpoint
--hls-manifests "[{\"ManifestName\":\"main\",\"ScteHls\":{\"AdMarkerHls\":
\\\"SCTE35_ENHANCED\\\"}}]"
```

API and SDK

To learn how to enable CUE tag ad markers for TS and CMAF origin endpoints via the MediaPackage live API or AWS SDK, see the following:

- [MediaPackage Live API reference](#)
- [AWS SDK](#)

Example HLS manifest showing SCTE-35 CUE tag signaling

This example shows a HLS manifest generated by MediaPackage using CUE tags to signal events in the live stream.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:8
#EXT-X-MEDIA-SEQUENCE:11
#EXTINF:3.3600001,
segment_h264_720p25_731013628.ts
#EXT-OATCLS-SCTE35:/DBhAAAAAAAAAAP/wBQb/tnK1YABLaK1DVUVJBAAAE3//
AAAPQPA0M0I6UDJJbm5pbmcxT3Z1cjMsUDpQMixU0k92ZXIsUzozLEk6MSxD01BSLTI1LTax0DM1MTQAAAAAF2kX6w==
#EXT-X-
ASSET:CAID=0x423A5032496E6E696E67314F766572332C503A50322C543A4F7665722C533A332C493A312C433A5052
```

```
#EXT-X-CUE-OUT:30.04
#EXTINF:3.36,
segment_h264_720p25_731013629.ts
#EXT-X-CUE-OUT-
CONT:ElapsedTime=3.36,Duration=30.04,CAID=0x423A5032496E6E696E67314F766572332C503A50322C543A4F7
DBhAAAAAAAAAAP/wBQb/tnKlYABLaKlDVUVJBAAAE3//
AAApQPAOM0I6UDJJbm5pbmcxT3ZlcjMsUDpQMixU0k92ZXIsUzozLEk6MSxD0lBSLTI1LTaxODM1MTQAAAAAF2kX6w==
#EXTINF:2.4,
segment_h264_720p25_731013630.ts
#EXT-X-CUE-OUT-
CONT:ElapsedTime=5.76,Duration=30.04,CAID=0x423A5032496E6E696E67314F766572332C503A50322C543A4F7
DBhAAAAAAAAAAP/wBQb/tnKlYABLaKlDVUVJBAAAE3//
AAApQPAOM0I6UDJJbm5pbmcxT3ZlcjMsUDpQMixU0k92ZXIsUzozLEk6MSxD0lBSLTI1LTaxODM1MTQAAAAAF2kX6w==
#EXTINF:2.4,
segment_h264_720p25_731013631.ts
...
#EXT-X-CUE-OUT-
CONT:ElapsedTime=27.36,Duration=30.04,CAID=0x423A5032496E6E696E67314F766572332C503A50322C543A4F7
DBhAAAAAAAAAAP/wBQb/tnKlYABLaKlDVUVJBAAAE3//
AAApQPAOM0I6UDJJbm5pbmcxT3ZlcjMsUDpQMixU0k92ZXIsUzozLEk6MSxD0lBSLTI1LTaxODM1MTQAAAAAF2kX6w==
#EXTINF:2.68,
segment_h264_720p25_731013640.ts
#EXT-OATCLS-SCTE35:/DBaAAAAAAAAAAP/wBQb/tpyBAABEAkJDVUVJBAAAE3+/
DjNC0lAySW5uaW5nMU92ZXIzLFA6UDIsVDpPdmVyLFM6MyxJ0jEsQzpQUi0yNS0wMTgzNTEAAD7WvT6
#EXT-X-CUE-IN
#EXTINF:2.4,
segment_h264_720p25_731013641.ts
#EXT-X-ENDLIST
```

DASH ad markers in AWS Elemental MediaPackage

DASH EventStream elements are used to signal all SCTE-35 `time_signal` or `splice_insert` markers present in the source. When you configure the ad markers to use the XML signaling, all the SCTE-35 markers parameters will be described through multiple nested XML elements inside the Event element. When you configure the Ad markers to use the Binary signaling, all the SCTE-35 markers parameters will be described through a Signal element including a binary base64-encoded representation of the SCTE marker, nested inside the Event element.

For more information about the XML and Binary signaling schemes for DASH, see section 5.5.2 of the [DASH Industry Forum IOP Guidelines v5](#).

The following table describes how to enable DASH ad markers in MediaPackage.

Console

To enable DASH ad markers when creating or editing an origin endpoint, in the MediaPackage console, under the DASH manifest settings, **SCTE configuration, Ad markers**, choose **XML** or **Binary**.

AWS CLI

To enable DASH ad markers for your origin endpoint, run one of the following commands in the AWS CLI replacing *region* with your own information:

```
aws --endpoint=https://mediapackagev2.region.amazonaws.com mediapackage --region
region create-origin-endpoint --channel-id test_channel --id test
_endpoint --dash-package "{\"AdMarkerDash\":\"XML\"}"
```

or

```
aws --endpoint=https://mediapackagev2.region.amazonaws.com mediapackage --region
region create-origin-endpoint --channel-id test_channel --id test
_endpoint --dash-package "{\"AdMarkerDash\":\"BINARY\"}"
```

API and SDK

To learn how to enable DASH ad markers for DASH manifests on CMAF origin endpoints via the MediaPackage live API or AWS SDK, see the following:

- [MediaPackage Live API reference](#)
- [AWS SDK](#)

Example DASH manifest showing SCTE-35 EXT-X-DATERANGE signaling

This example shows a DASH manifest generated by MediaPackage using XML signal approach to signal a `splice_insert` event in the live stream.

```
<MPD id="0" profiles="urn:mpeg:dash:profile:isoff-live:2011"
type="dynamic" minimumUpdatePeriod="PT2S" minBufferTime="PT5S"
timeShiftBufferDepth="PT3M28.267S" suggestedPresentationDelay="PT10S"
availabilityStartTime="2024-01-01T00:00:00.000000+00:00" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xmlns:scte35="urn:scte:scte35:2013:xml"
xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:mspr="urn:microsoft:playready"
```

```

xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/
ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
publishTime="2024-04-06T15:19:18.000806+00:00">
  <Period id="1712416567334" start="PT2319H16M7S" duration="PT30S">
    <EventStream schemeIdUri="urn:scte:scte35:2013:xml" timescale="90000">
      <Event presentationTime="2013000" duration="2700000">
        <scte35:SpliceInfoSection duration="2700000" protocolVersion="0"
ptsAdjustment="207000" tier="4095">
          <scte35:SpliceInsert spliceEventId="99" spliceEventCancelIndicator="false"
outOfNetworkIndicator="true" spliceImmediateFlag="false" uniqueProgramId="1"
availNum="1" availsExpected="1">
            <scte35:Program>
              <scte35:SpliceTime ptsTime="1806000"/>
            </scte35:Program>
            <scte35:BreakDuration autoReturn="true" duration="2700000"/>
          </scte35:SpliceInsert>
        </scte35:SpliceInfoSection>
      </Event>
    </EventStream>
    <AdaptationSet id="415376840" contentType="video" mimeType="video/mp4"
segmentAlignment="true">
      <SegmentTemplate timescale="60000" media="segment_${RepresentationID}_${Number
$.mp4" initialization="segment_${RepresentationID}_0_init.mp4" startNumber="6"
presentationTimeOffset="1342000">
        <SegmentTimeline>
          <S t="1342000" d="56000"/>
          <S t="1398000" d="240000" r="6"/>
          <S t="3078000" d="60000"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="1" bandwidth="200000" frameRate="30/1" codecs="avc1.42C01E"
width="360" height="240"/>
    </AdaptationSet>
    <AdaptationSet id="415406662" contentType="video" mimeType="video/mp4"
segmentAlignment="true">
      <SegmentTemplate timescale="60000" media="segment_${RepresentationID}_${Number
$.mp4" initialization="segment_${RepresentationID}_0_init.mp4" startNumber="6"
presentationTimeOffset="1342000">
        <SegmentTimeline>
          <S t="1342000" d="56000"/>
          <S t="1398000" d="240000" r="6"/>
          <S t="3078000" d="60000"/>
        </SegmentTimeline>
      </SegmentTemplate>

```

```

    <Representation id="2" bandwidth="400000" frameRate="30/1" codecs="avc1.42C01E"
width="480" height="360"/>
  </AdaptationSet>
  <AdaptationSet id="972192134" contentType="audio" mimeType="audio/mp4"
segmentAlignment="true" lang="und">
    <Label>und</Label>
    <SegmentTemplate timescale="48000" media="segment_${RepresentationID}_${Number
$.mp4" initialization="segment_${RepresentationID}_0_init.mp4" startNumber="6"
presentationTimeOffset="1073600">
      <SegmentTimeline>
        <S t="1076032" d="44032"/>
        <S t="1120064" d="192512"/>
        <S t="1312576" d="191488"/>
        <S t="1504064" d="192512"/>
        <S t="1696576" d="191488"/>
        <S t="1888064" d="192512"/>
        <S t="2080576" d="191488"/>
        <S t="2272064" d="192512"/>
        <S t="2464576" d="48128"/>
      </SegmentTimeline>
    </SegmentTemplate>
    <Representation id="3" bandwidth="98709" codecs="mp4a.40.2"
audioSamplingRate="48000">
      <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
    </Representation>
    <Representation id="4" bandwidth="98709" codecs="mp4a.40.2"
audioSamplingRate="48000">
      <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
    </Representation>
  </AdaptationSet>
  <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
value="2024-04-06T15:16:07.334Z"/>
</Period>
</MPD>

```

This example shows a DASH manifest generated by MediaPackage using Binary signal approach to signal a `splice_insert` event in the live stream.

```

<MPD id="0" profiles="urn:mpeg:dash:profile:isoff-live:2011"
type="dynamic" minimumUpdatePeriod="PT2S" minBufferTime="PT5S"

```

```

timeShiftBufferDepth="PT54.400S" suggestedPresentationDelay="PT10S"
availabilityStartTime="2024-01-01T00:00:00.000000+00:00" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xmlns:scte35="urn:scte:scte35:2013:xml"
xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:mspr="urn:microsoft:playready"
xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/
ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
publishTime="2024-04-12T17:44:20.000866+00:00">
  <Period id="1712943821467" start="PT2465H43M41S" duration="PT30S">
    <EventStream schemeIdUri="urn:scte:scte35:2014:xml+bin" timescale="90000">
      <Event presentationTime="2013000" duration="2700000">
        <Signal xmlns="http://www.scte.org/schemas/35/2016">
          <Binary>/DALAAAAAyiYAP/wFAUAAABjA0CAABu0sIAAKTLgAAEBAQAAbK+RAw==</Binary>
        </Signal>
      </Event>
    </EventStream>
    <AdaptationSet id="415376840" contentType="video" mimeType="video/mp4"
segmentAlignment="true">
      <SegmentTemplate timescale="60000" media="segment_${RepresentationID}_${Number
$.mp4" initialization="segment_${RepresentationID}_0_init.mp4" startNumber="5"
presentationTimeOffset="1342000">
        <SegmentTimeline>
          <S t="1342000" d="56000"/>
          <S t="1398000" d="240000" r="6"/>
          <S t="3078000" d="60000"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="1" bandwidth="200000" frameRate="30/1" codecs="avc1.42C01E"
width="360" height="240"/>
    </AdaptationSet>
    <AdaptationSet id="415406662" contentType="video" mimeType="video/mp4"
segmentAlignment="true">
      <SegmentTemplate timescale="60000" media="segment_${RepresentationID}_${Number
$.mp4" initialization="segment_${RepresentationID}_0_init.mp4" startNumber="5"
presentationTimeOffset="1342000">
        <SegmentTimeline>
          <S t="1342000" d="56000"/>
          <S t="1398000" d="240000" r="6"/>
          <S t="3078000" d="60000"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="2" bandwidth="400000" frameRate="30/1" codecs="avc1.42C01E"
width="480" height="360"/>
    </AdaptationSet>

```

```

    <AdaptationSet id="972192134" contentType="audio" mimeType="audio/mp4"
segmentAlignment="true" lang="und">
      <Label>und</Label>
      <SegmentTemplate timescale="48000" media="segment_${RepresentationID}_${Number}
$.mp4" initialization="segment_${RepresentationID}_0_init.mp4" startNumber="5"
presentationTimeOffset="1073600">
        <SegmentTimeline>
          <S t="1073856" d="44032"/>
          <S t="1117888" d="192512"/>
          <S t="1310400" d="191488"/>
          <S t="1501888" d="192512"/>
          <S t="1694400" d="191488"/>
          <S t="1885888" d="192512"/>
          <S t="2078400" d="191488"/>
          <S t="2269888" d="192512"/>
          <S t="2462400" d="48128"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="3" bandwidth="96446" codecs="mp4a.40.2"
audioSamplingRate="48000">
        <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
      <Representation id="4" bandwidth="96446" codecs="mp4a.40.2"
audioSamplingRate="48000">
        <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
      </Representation>
    </AdaptationSet>
    <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
value="2024-04-12T17:43:41.467Z"/>
  </Period>
</MPD>

```

Time-shifted viewing with AWS Elemental MediaPackage

Time-shifted viewing is available with live workflows in AWS Elemental MediaPackage.

Time-shifted viewing means that viewers can start watching a live stream at a time earlier than "now," permitting them to join from the beginning a show that's already in progress or to watch a show that's already completed. Time-shifted viewing effectively makes live events available for viewing on demand. MediaPackage supports time-shifted viewing for content that's up to 336

hours (14 days) old. You can enable time-shifted viewing for some or all of this content by defining the **startover window** on the endpoint. Manifests from within the startover window can be time shifted.

The following steps describe time-shift options and how to set them up. In these steps, "now" is determined either by the program date time (PDT) present in the source content from the encoder or, if this PDT information is not included, by the MediaPackage ingest time of the most recent segment.

Important

When using time-shifted viewing, we recommend using consistent playback windows across player sessions, rather than generating a unique start or end time for each viewer. This yields better caching at the CDN, and will avoid running into potential throttling related to those requests, on the MediaPackage level.

Step 1: Set the startover window

To create time-shifted manifests, you must define the live content that will be made available to view on demand. In MediaPackage, this eligible content is identified in your startover window. The startover window can be up to 24 hours long. You can set the window on the endpoint object, either through the MediaPackage console or MediaPackage API.

You might notice that the manifest lags behind real time when you initially create a startover window on an endpoint. This is because MediaPackage starts filling the manifest from the start of the window, and works up to "now." So, if you have a 24-hour startover window, MediaPackage fills the manifest starting 24 hours ago and working up to "now."

Step 2: Choose time-shifted options

You can apply time-shifted options to all manifests that originate from a MediaPackage endpoint by configuring the option on the endpoint. Alternatively, you can apply these options dynamically at the time of the content playback request by including query parameters.

You can choose from the following time-shifted options:

- **Delay content availability:** Set a delay for availability of live content.

- Define a manifest start and end: Set the time blocks of content are available from within the startover window. These blocks are defined by times provided in the start and end parameters.
- Limit a manifest duration : blocks of content with the specified duration are available within the startover window. The blocks are defined by the duration (in seconds) of the content and can come from different places in the startover window.

To use time shifting on all manifests that originate from an endpoint, set the filter configuration in the manifest settings of the endpoint. For more information, see [Manifest fields](#).

To use time shifting dynamically across requests, append query parameters to playback requests. For more information, see [Time-shifted query parameters](#).

Learn more about these options in the following topics.

Topics

- [Time-shifted query parameters in AWS Elemental MediaPackage](#)
- [Delay content availability from AWS Elemental MediaPackage](#)
- [Define manifest start and end times from AWS Elemental MediaPackage](#)
- [Limit manifest duration from AWS Elemental MediaPackage](#)
- [Time-shifted viewing examples in AWS Elemental MediaPackage](#)

Time-shifted query parameters in AWS Elemental MediaPackage

To use time-shifted viewing query parameters, append `aws.manifestsettings` to your playback request to MediaPackage. MediaPackage evaluates the query, and serves a client manifest based on those query parameters.

The following sections describe how to configure time-shift viewing query parameters.

Query parameter formatting

Use the following guidelines when constructing query parameters:

- Queries are not case sensitive.
- Queries can be up to 1,024 characters.
- Reserved characters in the queries must be URL encoded as indicated in the [URI: General Syntax](#) standard.

⚠ Important

At a minimum, if your query includes multiple parameters, the following characters in the query must be URL encoded. If they're not, MediaPackage processes just the first query parameter in the string.

Character	Encoded value
: (colon)	%3A
; (semicolon)	%3B
, (comma)	%2C
+ (plus)	%2B

If the query is malformed, MediaPackage returns an incomplete or empty manifest. For query syntax, see the following section.

Query syntax

The base query parameter for time-shifted viewing is `aws.manifestsettings`, which is followed by optional parameter name and value pairs. To construct the query, append `?aws.manifestsettings=` to the end of the MediaPackage endpoint URL, followed by parameter names and values. For a list of all of the available parameters, see [Query parameters](#).

An Apple HLS filter query might look like this:

```
https://example-mediapackage-endpoint.mediapackage.us-west-2.amazonaws.com/out/v1/examplemediapackage/index.m3u8?aws.manifestsettings=
```

The query syntax is listed in the following table.

ℹ Note

If you use Amazon CloudFront as your CDN, you might need to set additional configurations. For more information, see [Configure cache behavior for all endpoints](#).

Query string component	Description
?	A restricted character that marks the beginning of a query.
aws.manifestsetting=	The base query, which is followed by parameters constructed of name and value pairs. For a list of all of the available parameters, see Query parameters .
:	Used to associate the parameter name with a value. For example, <i>parameter_name :value</i> .
;	Separates parameters in a query that contains multiple parameters. For example, <i>parameter1_name:value ;parameter2_name:minValue-maxValue</i> . When used in a list of parameters for the same query, implies an AND operation.

Date and time format requirements

In all cases, the date and time must be notated in one of the following formats:

- ISO 8601 dates, such as 2017-08-18T21:18:54%2B08:00, where **%2B08:00** is the timezone UTC +08:00. The plus (+) must be URL-encoded.
- POSIX (or Epoch) time, such as 1503091134

DASH parameter rules

Start and end parameters in the URL request for DASH content can use standard parameter notation, or can be included as path elements in the URL.

- Query parameter notation – start and end parameters are included at the end of the request URL

Example

```
https://cf98fa7b2ee4450e.mediapackagev2.us-east-1.amazonaws.com/out/v1/997cbb27697d4863bb65488133bff26f/sports.mpd?start=1513717228&end=1513720828
```

- Path elements – start and end parameters are included in the path of the request URL

Example

```
https://cf98fa7b2ee4450e.mediapackagev2.us-east-1.amazonaws.com/out/  
v1/997cbb27697d4863bb65488133bff26f/start/2017-12-19T13:00:28-08:00/end/  
2017-12-19T14:00:28-08:00/sports.mpd
```

TS and CMAF parameter rules

Start and end parameters in the URL request for TS content can use standard parameter notation.

- Query parameter notation – start and end parameters are included at the end of the request URL

Example TS

```
https://cf98fa7b2ee4450e.mediapackagev2.us-east-1.amazonaws.com/out/  
v1/064134724fd74667ba294657a674ae72/  
comedy.m3u8?start=2017-12-19T13:00:28-08:00&end=2017-12-19T14:00:28-08:00
```

Example CMAF

```
https://cf98fa7b2ee4450e.mediapackagev2.us-east-1.amazonaws.com/out/  
v1/064134724fd74667ba294657a674ae72/manifest_id/  
news.m3u8?start=2018-04-04T01:14:00-08:00&end=2018-04-04T02:15:00-08:00
```

Query parameters

Playback devices can append the following time-shifted viewing query parameters to their requests to MediaPackage. Responses to requests with one or more of these parameters will include manifests that adhere to the specified time-shifted query parameters.

If you want all manifests served from this origin endpoint to be time shifted, enable the settings on the endpoint, as described in [Creating an origin endpoint in AWS Elemental MediaPackage](#).

Note

The `manifest_window_seconds` option is available only as a query parameter on a session initialization URL, and not as a setting on the endpoint. This is because the `manifest_window_seconds` query parameter modifies the standard length of the

manifest that's configured through the **Manifest window** setting on the endpoint. For more information about manifest window relationships, see [Window duration](#).

Name	Description	Example
clip_start_time	- With a time delay manifest, defines the earliest content that should be available in this manifest. Accepted values: Epoch or ISO 8601 timestamp that represents the earliest date and time that live content is available in the manifest. - clip_start_time can't be combined with start or end parameters. - clip_start_time query parameters can't be combined with Clip start time settings on the origin endpoint.	stream.mpd?aws.manifestsettings=subtitle_language=en-US,hi;clip_start_time:1720246778
manifest_window_seconds	- Used to request a manifest that's shorter than manifest window for this request. Accepted values: Integer that represents the desired manifest window in seconds. - At minimum, the value provided must be greater than or equal to double the length of the segment target duration. This is to ensure that the window includes at least two segments. - At maximum, the value provided must be less than the configured	stream.mpd?aws.manifestsettings=manifest_window_seconds:30

Name	Description	Example
	manifest window in the manifest settings for the origin endpoint.	
start	- Used to denote the beginning of a time-shifted manifest. For more information, see Define manifest start and end times from AWS Elemental MediaPackage. Accepted values: Epoch or ISO 8601 timestamp that represents the start date and time for content in the manifest.	stream.mpd?aws.manifestsettings=start:1732285823
end	- Used to denote the end of a time-shifted manifest. For more information, see Define manifest start and end times from AWS Elemental MediaPackage. Accepted values: Epoch or ISO 8601 timestamp that represents the start date and time for content in the manifest.	stream.mpd?aws.manifestsettings=end:1732300175
time_delay	Used to redefine the live point for this request. For more information, see Define manifest start and end times from AWS Elemental MediaPackage.	stream.mpd?aws.manifestsettings=time_delay:320

URL encoding query parameters

When you add query parameters to your HLS and LL-HLS playback requests, MediaPackage appends the parameters in alphabetical order to the child manifest URLs. For instance, the end parameter is listed before the start parameter.

To [create a signed API request](#) for AWS Signature Version 4 (SigV4) signing, your query strings must be URL encoded. By default, MediaPackage doesn't encode query parameters. To be a valid SigV4 signature, choose **URL-encode HLS child manifest query parameters** on your HLS or LL-HLS origin endpoint, as described in [Manifest fields](#).

For more information about SigV4 and its requirements, see [AWS Signature Version 4 for API requests](#) in *AWS Identity and Access Management User Guide*.

Example unencoded query strings

```
hls_video-1080.m3u8?  
aws.manifestsettings=time_delay:10&end=2025-02-03T17:09:49Z&start=2025-02-03T17:08:49Z
```

Example URL-encoded query strings

```
hls_video-1080.m3u8?aws.manifestsettings=time_delay  
%3A10&end=2025-02-03T17%3A09%3A49Z&start=2025-02-03T17%3A08%3A49Z
```

Delay content availability from AWS Elemental MediaPackage

You can specify a duration (in seconds) for MediaPackage to delay when content is available to players. The minimum time is 0 seconds. The following rules determine the maximum time:

- If `startoverwindow` is equal to 0, the maximum time is 86,400 seconds (24 hours).
- If `startoverwindow` is not equal to 0, the maximum time is the value of `startoverwindow`.

Use `time_delay` to redefine the live point and make content available at a time that equals "now" minus the delay specified. With a 60-second `time_delay`, content that MediaPackage receives at 12:20 isn't available until 12:21. Requests for playback at 12:20 will be served with content from 12:19. Likewise, if you're serving content across time zones, you can set a `time_delay` equal to the time zone difference to make content available at, for example, 8:00 local time.

The time delay duration must be less than the startover window duration.

Tip

Use a `time_delay` to help reduce buffering during input switching when you're using input redundancy with short output segments. Note that the delay can increase latency in content playback.

Using clip start with time delay

When you use the time delay function in MediaPackage, content availability is delayed by the specified amount. As content progresses, the manifest grows to your specified manifest window (seconds). Until the manifest has reached that window size with new content, content from before the start time can be available.

For example, one event ends at 925, and the other starts at 930. You set the time delay for the 930 event so that content is available at 932, and you defined your manifest window as 900 seconds (15 minutes). A viewer starts watching the event at 935, which is 3 minutes into the event. The manifest is 15 minutes long, so there is still another 12 minutes of content available from before the event started, so the viewer can see content from the event that ended at 925.

Depending on your contractual obligations, you might not want viewers to have access to the earlier content. To mitigate these concerns, you can use the clip start functionality in MediaPackage. With clip start, you can specify the earliest available content in the manifest. In the previous example, you could set the clip start time to 930 to ensure that viewers can't see content from before the current event.

You can enable the clip start setting on this endpoint so that all manifests that originate from this endpoint will include the clipped time. For information about adding clip start to an endpoint, see [Creating an origin endpoint in AWS Elemental MediaPackage](#).

Alternatively, manifest requests that include the `aws.manifestsettings=clip_start_time:` query parameter will be clipped so that content from before the specified time is not available. Requests to this endpoint that don't include the query parameter will receive a standard manifest, as defined by the endpoint's settings. MediaPackage returns an error to the requesting device if the clip start is set on the endpoint and included in request query parameters.

A note about start parameters

Clip start can't be used with *start* and *end* parameters.

Clip start and start parameters offer similar functionality, but have different practical purposes.

- Clip start is used to ensure that content before the set time is blocked. The manifest continues to grow, up to the defined window duration.
- Start is used to ensure that content before the set time is blocked. If used with an *end* parameter, the returned manifest ends at the specified time. If not used with an *end* parameter, the returned manifest continues to grow up to the value of the startover window.

Define manifest start and end times from AWS Elemental MediaPackage

MediaPackage accepts requests for up to 24 hours of content.

When requests with start and end parameters that are within the startover window are sent to this endpoint, MediaPackage generates a manifest for the requested timeframe. If the start parameter is outside of the startover window, or if the end parameter is before the startover window, the playback request fails. If no start and end parameters are used, the service generates a standard manifest.

Start and end parameters denote the beginning and end of a time-shifted manifest from MediaPackage. The playback device can append parameters to the end of a manifest request. Alternatively, the start and end parameters can be specified for the manifest through Filter Configuration parameters. For more information, see step 7 in [Manifest fields](#).

For packager-specific rules about how you can notate the parameters, see [Time-shifted query parameters](#).

The start and end parameters determine the time boundaries of the manifest. These are the expected behaviors based on request start and end parameters:

- If both start and end parameters are specified, the resulting manifest has a fixed start and end time that correspond to the specified start and end parameters.
- If the end time is in the future, the tags in the manifest are consistent with an event manifest. The manifest will continue to grow until it reaches the end time, at which point it will become a VOD manifest.
- If start or end parameters are used that are before the first segment is ingested, or after the last segment is ingested, the playback duration won't match the manifest duration as requested through the query parameters.

- If the start time is specified and the value is earlier than when the first segment is ingested, MediaPackage uses the actual time that the first segment is ingested as the start of the manifest.
- If the end time is specified and is later than when the last segment is ingested, MediaPackage uses the actual time that the last segment is ingested as the end of the manifest.
- If a start parameter is specified without an end parameter, the resulting manifest will have a fixed start time corresponding to the specified start parameter, while the end of the manifest will grow as the live content progresses.

Note

For TS output, many playback devices start playback at the current time ("now"). To view the content from the actual start time of the playback window, viewers can seek back on the playback progress bar.

- If no start time or end time parameters are specified, the service will generate a standard manifest.

Limit manifest duration from AWS Elemental MediaPackage

To dynamically request shorter manifests than the manifest window that's configured on the endpoint, use the `?aws.manifestsettings=manifest_window_seconds:` query parameter. When used at session initialization, the resulting manifest will conform to the number of seconds that you specify in the parameter. With `manifest_window_seconds`, you can serve manifests that are a variety of lengths, all from the same endpoint.

If `aws.manifestsettings=manifest_window_seconds:` isn't included in the request query parameters, MediaPackage serves a manifest the length of the window that's configured through **Manifest window (sec)** in the manifest settings when the endpoint was created.

Query parameters in the `aws.manifestsettings` namespace provide additional session-level time manipulation options. See [Time-shifted query parameters](#) for an overview.

To filter session-level manifests by audio, video, and subtitle formats, see [Manifest filtering](#).

Time-shifted viewing examples in AWS Elemental MediaPackage

These are MediaPackage time-shifted viewing examples.

Start parameters use case

I need content to start playing from where the customer left off.

Solution: Include the start parameter in the playback request URL. Program your playback device to record the time that they customer stopped the stream, then use this timestamp as the value for the start parameter.

Example

```
...stream.mpd?aws.manifestsettings=start:1732285823
```

End parameters use case

I need to define when the live content ends. Content is available for playback after this point, but live content won't be added.

Solution: Ensure the **startover window** on the endpoint accurately reflects how long you want the live content to be available for playback (up to 14 days). In playback requests, include the end parameter and use the time that live content ends as the parameter value.

Example

```
...stream.mpd?aws.manifestsettings=end:1732300175
```

Manifest window use case

I need to serve manifests of varying length from the same manifest URL because my customers use a variety of devices that have different limitations on manifest window lengths.

Solution: Include the manifest_window_seconds parameter in playback request URLs. Use the length limitations from each device as the value of for the manifest_window_seconds parameter.

Example

```
...stream.mpd?aws.manifestsettings=manifest_window_seconds:30
```

```
...stream.mpd?aws.manifestsettings=manifest_window_seconds:120
```

Enabling trick-play in AWS Elemental MediaPackage

Trick-play, sometimes called trick mode, provides a visual cue to viewers as they rewind, fast-forward, or seek through content in a digital video player. This helps the person using the video player to visualize where they are in the content timeline.

MediaPackage supports the following trick-play types:

Supported trick-play types for live workflows

Streaming protocol	I-frame only	Image-based
HLS with TS segments	✓	✓
HLS with CMAF segments	✓	✓
DASH	✓	✓

The following sections describe how to enable trick play in MediaPackage.

Topics

- [Using I-frame playlists to enable in AWS Elemental MediaPackage trick-play](#)
- [Using image media playlists to enable trick-play in AWS Elemental MediaPackage](#)

Using I-frame playlists to enable in AWS Elemental MediaPackage trick-play

MediaPackage supports live trick-play by creating an I-frame playlist from an existing live stream. The I-frame playlist contains the I-frame only video segments that your player uses for the image thumbnails. For information about I-frame playlists, see the [HTTP Live Streaming 2nd Edition specification](#).

To use an I-frame playlist to enable trick-play for HLS

- In the MediaPackage console, choose **Include Iframe-only stream** in the Segment settings when creating or editing an endpoint. MediaPackage generates I-frame only streams for each rendition in the parent playlist.

This example HLS parent playlist generated by MediaPackage uses EXT-X-I-FRAME-STREAM-INF tags for each rendition and includes the "index_1_iframe.m3u8" and "index_2_iframe.m3u8" child playlist URIs.

```
#EXTM3U
#EXT-X-VERSION:4
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-STREAM-INF:CODECS="avc1.42001f,mp4a.40.2",AVERAGE-
BANDWIDTH=1000,RESOLUTION=1920x1080,VIDEO-RANGE=SDR,FRAME-
RATE=30.0,BANDWIDTH=2000,CLOSED-CAPTIONS="CC"
index_1.m3u8
#EXT-X-STREAM-INF:CODECS="avc1.42001f,mp4a.40.2",AVERAGE-
BANDWIDTH=2000,RESOLUTION=1920x1080,VIDEO-RANGE=SDR,FRAME-
RATE=30.0,BANDWIDTH=3000,CLOSED-CAPTIONS="CC"
index_2.m3u8
#EXT-X-STREAM-INF:CODECS="mp4a.40.2",AVERAGE-BANDWIDTH=1000,BANDWIDTH=2000
index_3.m3u8
#EXT-X-I-FRAME-STREAM-
INF:CODECS="avc1.42001f,mp4a.40.2",RESOLUTION=1920x1080,VIDEO-
RANGE=SDR,BANDWIDTH=1000,URI="index_1_iframe.m3u8"
#EXT-X-I-FRAME-STREAM-
INF:CODECS="avc1.42001f,mp4a.40.2",RESOLUTION=1920x1080,VIDEO-
RANGE=SDR,BANDWIDTH=1000,URI="index_2_iframe.m3u8"
```

MediaPackage also inserts EXT-I-FRAMES-ONLY tags in the child playlist, and then generates and includes an I-frame only playlist in the stream. This playlist enables player functionality like fast forward and rewind.

This example HLS child playlist generated by MediaPackage uses the EXT-X-I-FRAMES-ONLY tag and includes the index_1_iframe_0.ts segment URI.

```
#EXTM3U
#EXT-X-VERSION:4
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-DISCONTINUITY-SEQUENCE:0
#EXT-X-I-FRAMES-ONLY
#EXT-X-PROGRAM-DATE-TIME:2021-12-07T10:00:00Z
#EXTINF:6.0,
```

```
index_1_iframe_0.ts
```

To use an I-frame playlist to enable trick-play for DASH

- MediaPackage supports live trick-play by creating an I-frame representation from an existing live stream. The I-frame representation contains the I-frame only video segments that your player uses for the image thumbnails.

For information about I-frame playlists with DASH, see the [DASH-IF Interoperability Points specification, v4.3, section 3.2.9](#).

In the MediaPackage console, choose **Include Iframe-only stream** in the Segment settings, when creating or editing an endpoint. MediaPackage generates I-frame only streams for each rendition in the DASH manifest.

This example DASH manifest generated by MediaPackage uses a specific AdaptationSet that includes a `<EssentialProperty schemeIdUri="http://dashif.org/guidelines/trickmode" />` element and Representation elements including both a `frameRate` and a `maxPlayoutRate` attributes, which values are representative of trickplay track properties.

```
<MPD id="0" profiles="urn:mpeg:dash:profile:isoff-live:2011"
  type="dynamic" minimumUpdatePeriod="PT2S" minBufferTime="PT5S"
  timeShiftBufferDepth="PT1M4.559S" suggestedPresentationDelay="PT10S"
  availabilityStartTime="2024-01-01T00:00:00.000000+00:00"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:mspr="urn:microsoft:playready"
  xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/
  ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
  publishTime="2024-04-02T17:05:14.000894+00:00">
  <Period id="1712019690992" start="PT2209H1M30S">
    <AdaptationSet id="1170682507" contentType="video" mimeType="video/mp4"
      segmentAlignment="true" codingDependency="false">
      <EssentialProperty schemeIdUri="http://dashif.org/guidelines/trickmode"
        value="2137622408"/>
      <SegmentTemplate timescale="48000" media="cmaf-segment_${RepresentationID}
      ${_Number$.mp4" initialization="cmaf-segment_${RepresentationID}_8005_init.mp4"
        startNumber="23393" presentationTimeOffset="831120">
        <SegmentTimeline>
          <S t="6589849554" d="180179"/>
          <S t="6590029734" d="180179"/>
          <S t="6590209914" d="180179"/>
```

```

        <S t="6590390094" d="180179"/>
        <S t="6590570274" d="180179"/>
        <S t="6590750454" d="180179"/>
        <S t="6590930634" d="180179"/>
        <S t="6591110814" d="180179"/>
        <S t="6591290994" d="180179"/>
        <S t="6591471174" d="180179"/>
        <S t="6591651354" d="180179"/>
        <S t="6591831534" d="180179"/>
        <S t="6592011714" d="180179"/>
        <S t="6592191894" d="180179"/>
        <S t="6592372074" d="180179"/>
        <S t="6592552254" d="180179"/>
    </SegmentTimeline>
</SegmentTemplate>
    <Representation id="1_iframe" bandwidth="1048" frameRate="24000/90090"
codecs="avc1.4D401F" width="1280" height="720" maxPlayoutRate="90"/>
    <Representation id="2_iframe" bandwidth="1572" frameRate="24000/90090"
codecs="avc1.4D401F" width="1920" height="1080" maxPlayoutRate="90"/>
</AdaptationSet>
    <AdaptationSet id="2137622408" contentType="video" mimeType="video/mp4"
segmentAlignment="true">
        <SegmentTemplate timescale="48000" media="cmaf-segment_${RepresentationID}
_${Number$.mp4" initialization="cmaf-segment_${RepresentationID}_8005_init.mp4"
startNumber="23393" presentationTimeOffset="831120">
            <SegmentTimeline>
                <S t="6589849554" d="180179"/>
                <S t="6590029734" d="180179"/>
                <S t="6590209914" d="180179"/>
                <S t="6590390094" d="180179"/>
                <S t="6590570274" d="180179"/>
                <S t="6590750454" d="180179"/>
                <S t="6590930634" d="180179"/>
                <S t="6591110814" d="180179"/>
                <S t="6591290994" d="180179"/>
                <S t="6591471174" d="180179"/>
                <S t="6591651354" d="180179"/>
                <S t="6591831534" d="180179"/>
                <S t="6592011714" d="180179"/>
                <S t="6592191894" d="180179"/>
                <S t="6592372074" d="180179"/>
                <S t="6592552254" d="180179"/>
            </SegmentTimeline>
        </SegmentTemplate>
    </AdaptationSet>

```

```

    <Representation id="1" bandwidth="2000000" frameRate="24000/1001"
    codecs="avc1.4D401F" width="1280" height="720"/>
    <Representation id="1" bandwidth="5000000" frameRate="24000/1001"
    codecs="avc1.4D401F" width="1920" height="1080"/>
  </AdaptationSet>
  <AdaptationSet id="972192134" contentType="audio" mimeType="audio/mp4"
  segmentAlignment="true" lang="und">
    <Label>und</Label>
    <SegmentTemplate timescale="48000" media="cmf-segment_${RepresentationID}
    ${_Number}.mp4" initialization="cmf-segment_${RepresentationID}_8005_init.mp4"
    startNumber="23393" presentationTimeOffset="831120">
      <SegmentTimeline>
        <S t="6589853020" d="180224" r="1"/>
        <S t="6590213469" d="180224" r="1"/>
        <S t="6590573917" d="179200"/>
        <S t="6590753117" d="180224"/>
        <S t="6590933342" d="180224" r="3"/>
        <S t="6591654239" d="180224" r="1"/>
        <S t="6592014688" d="180224" r="3"/>
      </SegmentTimeline>
    </SegmentTemplate>
    <Representation id="6" bandwidth="191522" codecs="mp4a.40.2"
    audioSamplingRate="48000">
      <AudioChannelConfiguration
      schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
    </Representation>
  </AdaptationSet>
  <SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
  value="2024-04-02T17:04:13.680Z"/>
</Period>
</MPD>

```

Using image media playlists to enable trick-play in AWS Elemental MediaPackage

To use image-based trickplay, in your upstream encoder you create an HLS *image media playlist* that contains JPEG image segments. MediaPackage automatically passes through the image segments to the output. These segments are the thumbnail images and image metadata that the video player uses for visual cues. These segments must conform to the [Image Media Playlist specification, version 0.4](#). The service supports the time-based implementation of the specification.

For information about how to configure your upstream encoder to generate an image media playlist, see [Configuring your upstream encoder to generate image media playlists](#).

Input source requirements

Your HLS source content must meet the following requirements:

- The HLS parent playlist that references the image playlist must include the EXT-X-IMAGE-STREAM-INF tag.
- The image playlist must include An EXT-X-IMAGES-ONLY tag above the segment list.

Note

We recommend that you use decimal durations in the EXT-INF and EXT-X-TILES tags to help MediaPackage give players the most accurate image durations.

- You must use image segments that are valid JPEG image files less than 20 MB.
- Each JPEG must contain only one image segment. The encoder must produce image segments and video segments at the same cadence.

You can use AWS Media Services to generate an HLS source in your upstream encoder that complies with the [Image Media Playlist specification, version 0.4](#). For more information, see the following section [Configuring your upstream encoder to generate image media playlists](#).

Limitations

Keep in mind the following limitations when using image-based trick-play for MediaPackage:

- MediaPackage doesn't combine image segments for packaging configurations. For example, if the service ingests a live stream with an image asset with a 2 second segment duration, and you specify a segment output duration of 6 seconds, we combine the video and audio segments to be 6 seconds long, but image segments will remain 2 seconds.
- Depending on your HLS player requirements, the use of EXT-X-PROGRAM-DATE-TIME tags might be necessary to display the trick-play image.

Configuring your upstream encoder to generate image media playlists

Your HLS source must conform to the [Image Media Playlist specification, version 0.4](#). You can use the following AWS Media Services to create an HLS stream that complies with the specification. For more information, see the following documentation:

- [Trick-play track via the Image Media Playlist specification](#) in the *Elemental Live User Guide*.
- [Trick-play track via the Image Media Playlist specification](#) in the *AWS Elemental MediaLive User Guide*.

To use an image media playlist to enable trick-play for HLS

- MediaPackage supports live trick-play by creating an image media playlist from an existing live stream. Your source HLS content must conform to the [Image Media Playlist specification, version 0.4](#).

If your source content includes one or more image media playlists, your HLS output will also include the same image media playlists, with the corresponding signaling in the parent playlist.

This example HLS parent playlist generated by MediaPackage uses a EXT-X-IMAGE-STREAM-INF tag for the image-based trickplay rendition and includes the "variant-hls_6.m3u8" child playlist URI. child playlist URIs.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-
MEDIA:LANGUAGE="und",AUTOSELECT=YES,CHANNELS="2",FORCED=NO,TYPE=AUDIO,URI="variant-
hls_5.m3u8",GROUP-ID="audio_0",DEFAULT=YES,NAME="und"
#EXT-X-STREAM-INF:CODECS="mp4a.40.2,avc1.4D4028",AVERAGE-
BANDWIDTH=5500000,RESOLUTION=1920x1080,VIDEO-RANGE=SDR,FRAME-
RATE=30.0,BANDWIDTH=5931112,AUDIO="audio_0"
variant-hls_1.m3u8
#EXT-X-STREAM-INF:CODECS="mp4a.40.2,avc1.4D401F",AVERAGE-
BANDWIDTH=3300000,RESOLUTION=1280x720,VIDEO-RANGE=SDR,FRAME-
RATE=30.0,BANDWIDTH=3643112,AUDIO="audio_0"
variant-hls_2.m3u8
#EXT-X-STREAM-INF:CODECS="mp4a.40.2,avc1.4D401F",AVERAGE-
BANDWIDTH=1650000,RESOLUTION=960x540,VIDEO-RANGE=SDR,FRAME-
RATE=30.0,BANDWIDTH=1927083,AUDIO="audio_0"
variant-hls_3.m3u8
```

```
#EXT-X-STREAM-INF:CODECS="mp4a.40.2, avc1.4D401E", AVERAGE-
BANDWIDTH=825000, RESOLUTION=640x360, VIDEO-RANGE=SDR, FRAME-
RATE=30.0, BANDWIDTH=1069069, AUDIO="audio_0"
variant-hls_4.m3u8
#EXT-X-IMAGE-STREAM-
INF:CODECS="jpeg", RESOLUTION=312x176, BANDWIDTH=131436, URI="variant-hls_6.m3u8"
```

MediaPackage also inserts EXT-X-IMAGES-ONLY tags in the child playlist, and then generates and includes an image media playlist in the stream. This playlist enables player functionality like fast forward and rewind.

This example HLS child playlist generated by MediaPackage uses the EXT-X-IMAGES-ONLY tag and includes the segment_6_4595218.jpg segment URI.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:4595218
#EXT-X-DISCONTINUITY-SEQUENCE:22
#EXT-X-IMAGES-ONLY
#EXT-X-PROGRAM-DATE-TIME:2024-04-18T19:16:56.833Z
#EXTINF:6.0,
segment_6_4595218.jpg
#EXT-X-PROGRAM-DATE-TIME:2024-04-18T19:17:02.833Z
#EXTINF:6.0,
segment_6_4595219.jpg
#EXT-X-PROGRAM-DATE-TIME:2024-04-18T19:17:08.833Z
#EXTINF:6.0,
segment_6_4595220.jpg
#EXT-X-PROGRAM-DATE-TIME:2024-04-18T19:17:14.833Z
#EXTINF:6.0,
segment_6_4595221.jpg
#EXT-X-PROGRAM-DATE-TIME:2024-04-18T19:17:20.833Z
```

To use an image media playlist to enable trick-play for DASH

- MediaPackage supports live trick-play by creating an image media playlist from an existing live stream. Your source HLS content must conform to the [Image Media Playlist specification, version 0.4](#).

If your source content includes one or more image media playlists, your DASH output will also include an image-based AdaptationSet and the corresponding image Representations. When MediaPackage outputs content from a DASH packaging configuration or endpoint, the service outputs thumbnails based on the [DASH-IF Interoperability Points specification, v4.3, section 6.2.6](#).

This example DASH manifest generated by MediaPackage uses a specific AdaptationSet in which the representation includes a `<EssentialProperty schemeIdUri="http://dashif.org/guidelines/thumbnail_tile" value="1x1"/>` element indicating that each image represents a discrete image, and not a tiled thumbnails image that includes multiple thumbnails.

```
<MPD id="0" profiles="urn:mpeg:dash:profile:isoff-live:2011"
  type="dynamic" minimumUpdatePeriod="PT2S" minBufferTime="PT5S"
  timeShiftBufferDepth="PT1M4.079S" suggestedPresentationDelay="PT10S"
  availabilityStartTime="2024-01-01T00:00:00.000000+00:00"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:mpeg:dash:schema:mpd:2011" xmlns:mspr="urn:microsoft:playready"
  xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 http://standards.iso.org/
  ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd"
  publishTime="2024-04-02T16:57:17.000063+00:00">
  <Period id="1712019690992" start="PT2209H1M30S">
    <AdaptationSet id="2137622408" contentType="video" mimeType="video/mp4"
      segmentAlignment="true">
      <SegmentTemplate timescale="48000" media="cmaf-segment_${RepresentationID}
      $_$Number$.mp4" initialization="cmaf-segment_${RepresentationID}_21187_init.mp4"
      startNumber="36447" presentationTimeOffset="831120">
        <SegmentTimeline>
          <S t="6566786514" d="180179"/>
          <S t="6566966694" d="180179"/>
          <S t="6567146874" d="180179"/>
          <S t="6567327054" d="180179"/>
          <S t="6567507234" d="180179"/>
          <S t="6567687414" d="180179"/>
          <S t="6567867594" d="180179"/>
          <S t="6568047774" d="180179"/>
          <S t="6568227954" d="180179"/>
          <S t="6568408134" d="180179"/>
          <S t="6568588314" d="180179"/>
          <S t="6568768494" d="180179"/>
          <S t="6568948674" d="180179"/>
        </SegmentTimeline>
      </SegmentTemplate>
    </AdaptationSet>
  </Period>
</MPD>
```

```

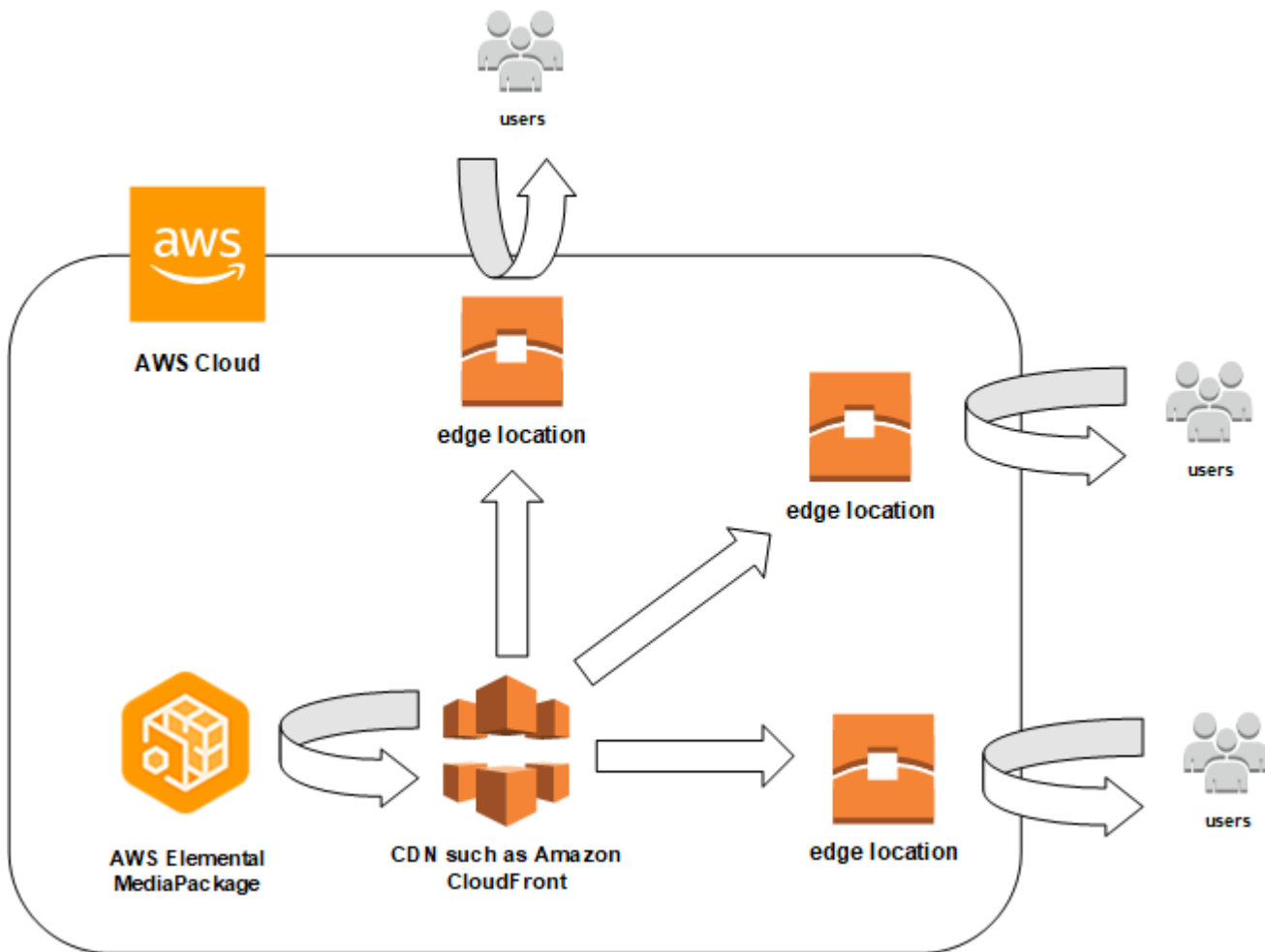
        <S t="6569128854" d="180179"/>
        <S t="6569309034" d="180179"/>
        <S t="6569489214" d="180179"/>
    </SegmentTimeline>
</SegmentTemplate>
<Representation id="1" bandwidth="2000000" frameRate="24000/1001"
codecs="avc1.4D401F" width="1280" height="720"/>
</AdaptationSet>
<AdaptationSet id="515700000" contentType="image" mimeType="image/jpeg"
segmentAlignment="true">
    <SegmentTemplate timescale="60000" media="cmaf-segment_${RepresentationID}
_${Number$.jpg" initialization="cmaf-segment_${RepresentationID$_21187_init.jpg"
startNumber="36447" presentationTimeOffset="1038900">
        <SegmentTimeline>
            <S t="8208483142" d="225224"/>
            <S t="8208708367" d="225224"/>
            <S t="8208933592" d="225224"/>
            <S t="8209158817" d="225224"/>
            <S t="8209384042" d="225224"/>
            <S t="8209609267" d="225224"/>
            <S t="8209834492" d="225224"/>
            <S t="8210059717" d="225224"/>
            <S t="8210284942" d="225224"/>
            <S t="8210510167" d="225224"/>
            <S t="8210735392" d="225224"/>
            <S t="8210960617" d="225224"/>
            <S t="8211185842" d="225224"/>
            <S t="8211411067" d="225224"/>
            <S t="8211636292" d="225224"/>
            <S t="8211861517" d="225224"/>
        </SegmentTimeline>
    </SegmentTemplate>
    <Representation id="3" bandwidth="28608" codecs="jpeg" width="320"
height="174">
        <EssentialProperty schemeIdUri="http://dashif.org/guidelines/
thumbnail_tile" value="1x1"/>
    </Representation>
</AdaptationSet>
<AdaptationSet id="972192134" contentType="audio" mimeType="audio/mp4"
segmentAlignment="true" lang="und">
    <Label>und</Label>
    <SegmentTemplate timescale="48000" media="cmaf-segment_${RepresentationID}
_${Number$.mp4" initialization="cmaf-segment_${RepresentationID$_21187_init.mp4"
startNumber="36447" presentationTimeOffset="831120">

```

```
<SegmentTimeline>
  <S t="6566792949" d="180224" r="1"/>
  <S t="6567153398" d="180224" r="3"/>
  <S t="6567874295" d="180224" r="1"/>
  <S t="6568234744" d="180224" r="3"/>
  <S t="6568955641" d="180224" r="2"/>
  <S t="6569496313" d="179200"/>
</SegmentTimeline>
</SegmentTemplate>
<Representation id="6" bandwidth="191522" codecs="mp4a.40.2"
audioSamplingRate="48000">
  <AudioChannelConfiguration
schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
  </Representation>
</AdaptationSet>
<SupplementalProperty schemeIdUri="urn:scte:dash:utc-time"
value="2024-04-02T16:56:13.200Z"/>
</Period>
</MPD>
```

Working with AWS Elemental MediaPackage and CDNs

You can use a content delivery network (CDN) such as [Amazon CloudFront](#) to serve the content that you store in AWS Elemental MediaPackage. A CDN is a globally distributed set of servers that caches content such as videos. When a user requests your content, the CDN routes the request to the edge location that provides the lowest latency. If your content is already cached in that edge location, the CDN delivers it immediately. If your content is not currently in that edge location, the CDN retrieves it from your origin (in this case, the MediaPackage endpoint) and distributes it to the user. The following illustration shows this process.



The following sections provide more information about using a CDN in your MediaPackage workflow.

Topics

- [CDN configuration recommendations](#)

- [Secure MediaPackage content with CDN authorization](#)

CDN configuration recommendations

To configure your CDN distributions for delivering Apple HLS and low-latency HLS (LL-HLS) streams with MediaPackage, we suggest using the following approach. Ensure that all your usual default parameters in the CDN distribution configuration are compatible with the delivery of HLS and LL-HLS streams with MediaPackage.

Honor MediaPackage 'cache-control: max-age' values

MediaPackage defines time-to-live (TTL) values for objects either statically or dynamically, depending on the object type. The specific TTLs are listed below, and it's strongly recommended that you honor these values and avoid overriding them at the CDN configuration level.

- Multivariant playlist (for both regular HLS and LL-HLS) - half the duration of the media segments
- Media playlists (regular HLS) - half the duration of the media segments
- Media playlists (LL-HLS) - 1 second
- TS media segments and init segments - 1209600 seconds (14 days)
- CMAF media segments and initialization segments - 1209600 seconds (14 days)

Include specific query strings in your CDN cache key

We recommend that you include the following query strings in the CDN cache key.

- `aws.manifestfilter`: used to customize the manifest contents, as described in [Manifest filtering](#).
- `aws.manifestsettings`: used to apply time-shifted settings to the manifest contents, as described in [Time-shifted viewing](#).
- `_HLS_msn` and `_HLS_part`: used to support LL-HLS playback request logic, as described in the *HTTP Live Streaming Documentation* [Enabling Low-Latency HTTP Live Streaming \(HLS\)](#).

MediaPackage ignores all other query strings. Don't include them in the CDN forward requests.

Response timeout

LL-HLS uses the Blocking Requests mechanism for both playlists and media parts, which is signaled through the EXT-X-PRELOAD-HINT tag. This mechanism puts the origin response on hold until the object is fully available. Consequently, the CDN should also wait for the origin response, and therefore, the response timeout value in your CDN distribution should be at least three times your parts duration.

Forwarded HTTP headers

You may consider including the Origin header in your CDN forward requests to MediaPackage as the only potentially necessary HTTP request header. In doing so, MediaPackage will respond with an `access-control-allow-origin` header, using the value passed as the Origin header value. If the Origin header is not included in the forward requests, MediaPackage will respond with an `access-control-allow-origin: *` header. Choose the approach that best fits your CORS requirements.

Forwarded cookies

MediaPackage doesn't consider any cookies that may be sent together with forward requests. Therefore, it's advisable to exclude cookies from CDN forward requests.

Secure MediaPackage content with CDN authorization

AWS Elemental MediaPackage CDN authorization helps you protect your streaming content from unauthorized access and direct origin requests. When you configure CDN authorization, MediaPackage only fulfills playback requests that include valid authorization headers from your content delivery network, preventing users from bypassing your CDN to access content directly.

If you use Amazon CloudFront for your CDN, you can configure access to MediaPackage resources with [AWS Signature Version 4 \(SigV4\) authentication](#).

If your CDN doesn't support SigV4, use the following instructions to set up authorization headers between your CDN and MediaPackage.

How it works

You configure your CDN to include a *custom HTTP header* in content requests to MediaPackage.

The custom HTTP header must use the exact name **X-MediaPackageV2-CDNIdentifier** with a value that is 8-256 characters long. We strongly recommend using the [UUID version 4](#) format for the value, which produces a 36-character string that is both unique and unpredictable.

Example header

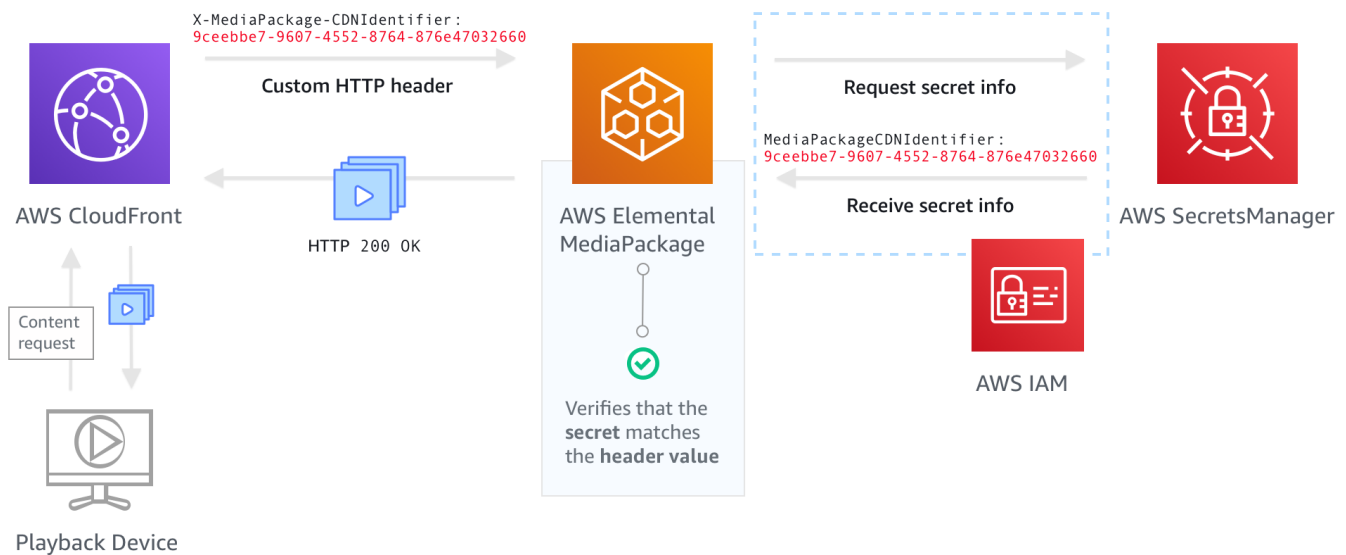
The following example shows the required header format.

```
X-MediaPackageV2-CDNIdentifier: 9ceebbe7-9607-4552-8764-876e47032660
```

You store the header value as a *secret* in AWS Secrets Manager. When your CDN sends a playback request, MediaPackage verifies the custom HTTP header value. MediaPackage compares this value with the stored secret. An AWS Identity and Access Management permissions policy and role grant MediaPackage permission to read the secret.

If the values match, MediaPackage serves the content along with an HTTP 200 OK status code. If the values don't match, or if the authorization request fails, MediaPackage doesn't serve the content and returns an HTTP 403 Unauthorized status code.

The following image shows successful CDN authorization using Amazon CloudFront.



Complete the following procedures to configure CDN authorization with MediaPackage.

Topics

- [Configure MediaPackage CDN authorization setup](#)
- [Rotate MediaPackage CDN authorization secrets](#)
- [Troubleshoot MediaPackage CDN authorization errors](#)
- [Optimize MediaPackage CDN authorization security](#)

Configure MediaPackage CDN authorization setup

Configure AWS Elemental MediaPackage CDN authorization to secure your streaming content by setting up custom HTTP headers, storing secrets, and configuring IAM permissions. This procedure ensures that only authorized CDN requests can access your MediaPackage endpoints.

To set up CDN authorization

1. **Configure a CDN custom origin HTTP header.** In your CDN, configure a custom origin HTTP header that contains the header **X-MediaPackageV2-CDNIdentifier** and a value. For the value, we recommend that you use the [UUID version 4](#) format, which produces a 36-character string. If you aren't using the UUID version 4 format, the value must be 8-256 characters long.

Important

The value you choose should be a static value. There isn't native integration between your CDN and AWS Secrets Manager, so the value should be static both in your CDN and in AWS Secrets Manager. If you change this value after you configure your CDN and your secret, you have to manually rotate the value. For more information, see [Rotate MediaPackage CDN authorization secrets](#).

Example header and value

```
X-MediaPackageV2-CDNIdentifier: 9ceebbe7-9607-4552-8764-876e47032660
```

These steps will vary depending on your CDN. For distribution setup with custom headers in Amazon CloudFront, see the [Distribution settings reference](#) in the Amazon CloudFront developer guide.

2. **Store the value as a secret in AWS Secrets Manager.** Store the custom header value as a *secret* in AWS Secrets Manager. The secret must use the same AWS account and Region

settings as your MediaPackage resources. MediaPackage doesn't support sharing secrets across accounts or Regions. However, you can use the same secret across multiple endpoints in the same Region and on the same account.

⚠ Important

The secret must be created in the same AWS Region as your MediaPackage endpoint. Cross-Region secret access is not supported.

- a. Sign in to the AWS Secrets Manager console at <https://console.aws.amazon.com/secretsmanager/>.
- b. Choose **Store a new secret**. For **Secret type**, choose **Other type of secrets**.
- c. For **Key/value pairs**, enter the key and value information.
 - In the box on the left (key), enter **MediaPackageV2CDNIdentifier**.
 - In the box on the right (value), enter the value that you configured for your custom origin HTTP header. For example, 9ceebbe7-9607-4552-8764-876e47032660.
- d. For **Encryption key**, you can use the AWS KMS key that Secrets Manager creates by default.
- e. Choose **Next**.
- f. For **Secret name**, we recommend that you prefix it with **MediaPackageV2/** so that you know it's a secret used for MediaPackage. For example, **MediaPackageV2/cdn_auth_us-west-2**.
- g. Choose **Next**.
- h. For **Configure automatic rotation**, keep the default **Disable automatic rotation** setting.

If you need to rotate the key later, see [Rotate MediaPackage CDN authorization secrets](#).
- i. Choose **Next**, and then choose **Store**.

This takes you to the list of your secrets.

- j. Select your secret name to view the **Secret ARN**. The ARN has a value similar to `arn:aws:secretsmanager:us-west-2:123456789012:secret:MediaPackageV2/cdn_auth_test-xxxxxx`. You use the ARN for the secret when you configure CDN authorization for MediaPackage in step 4.

3. **Create an IAM role for MediaPackage access to Secrets Manager.** Create an IAM role to give MediaPackage access to Secrets Manager and AWS Key Management Service (AWS KMS). When MediaPackage receives a playback request from the CDN that includes a secret value in the custom headers, MediaPackage retrieves the stored secret value from AWS KMS and verifies that the secret values match. Follow the steps in [the section called “Allowing MediaPackage to access other AWS services”](#) to set up the policy and role.

You use the ARN for the IAM role that you created when you enable CDN authorization in MediaPackage in the next step.

4. **Enable CDN authorization in MediaPackage.** You can enable CDN authorization for your channel endpoints from the MediaPackage console, AWS CLI, or MediaPackage API.

Tip

Use the same secret across multiple endpoints in the same Region and on the same account. Reduce costs by creating a new secret only when necessary for your workflow.

- a. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.
- b. Under **Live v2**, create or edit a channel group and channel. For help, see [Creating a channel group in AWS Elemental MediaPackage](#).
- c. On the channel, create or edit an endpoint.
- d. In **Endpoint policy**, select **Attach a custom policy** and add a policy for the endpoint.

Example policy with CDN auth condition

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGetObjectAccessForAuthorizedRequest",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "mediapackagev2:GetObject",
```

```

    "Resource": "arn:aws:mediapackagev2:us-
east-1:111122223333:channelGroup/channelGroupName/channel/channelName/
originEndpoint/originEndpointName",
    "Condition": {
        "Bool": {
            "mediapackagev2:RequestHasMatchingCdnAuthHeader": "true"
        }
    }
}
]
}

```

- e. In **CDN authorization configuration**, complete the fields:
 - In **Secrets role ARN**, enter the ARN for the IAM role that you created in step 3.
 - In **CDN identifier secret ARN**, enter the ARN for the secret in Secrets Manager that your CDN uses for authorization to access your endpoint.
- f. Complete the remaining fields as needed and save the endpoint.

You have now completed the setup for CDN authorization. Requests to this endpoint must contain the same authorization code that you saved in Secrets Manager.

To enable CDN authorization via the MediaPackage API

For information about enabling CDN authorization with the MediaPackage API, see [AWS Elemental MediaPackage V2 Live API Reference](#).

Rotate MediaPackage CDN authorization secrets

When you need to update your AWS Elemental MediaPackage CDN authorization credentials, you must rotate the stored secret value in AWS Secrets Manager to maintain synchronization with your CDN's custom HTTP header. This process ensures continuous content delivery while updating security credentials.

To rotate the value

1. Update the stored secret value in Secrets Manager as described in [Modifying a secret](#) in the *AWS Secrets Manager User Guide*.

To ensure continued playback for active streams, MediaPackage authorizes requests that use either the current value in Secrets Manager or one version back.

2. Wait 5 minutes for MediaPackage to recognize that the value has changed in Secrets Manager.
3. In your CDN, update the value in `X-MediaPackageV2-CDNIdentifier` to the new secret value.
4. Wait for your CDN to update fully with the new value before you send any requests through it to MediaPackage.

Emergency rotation details

In emergency situations where you need to immediately invalidate a secret, do one of the following:

- Update the endpoint configuration to remove the compromised secret ARN and replace it with a new secret ARN.
- Alternatively, rotate the secret twice in quick succession. After 5 minutes, MediaPackage will no longer accept the old secret values since the incoming header value would not match either the current or previous secret.

Note

MediaPackage caches secret values for 5 minutes. During this period, both the old and new secret values may be valid for authorization.

Troubleshoot MediaPackage CDN authorization errors

When AWS Elemental MediaPackage CDN authorization fails, you may encounter various error codes and authorization issues. This section helps you identify and resolve common problems with CDN authorization configuration, secret management, and IAM permissions.

Common Error Scenarios and Resolutions

Scenario	Error Type	Resolution
Secret validation failure	4XX error	Verify that your secret is stored with the correct key name <code>MediaPackageV2CDNI</code>

Scenario	Error Type	Resolution
		identifier and the value is between 8-256 characters.
IAM role access denied	4XX error	Check that the IAM role has the correct permissions and trust relationship as described in Configure MediaPackage CDN authorization setup .
Secret not found	4XX error	Verify that the secret ARN is correct and the secret exists in the same Region as your MediaPackage endpoint.
Header value mismatch	403 Unauthorized	Ensure that the value in the X-MediaPackageV2-CDNIdentifier header matches the value stored in Secrets Manager.

Optimize MediaPackage CDN authorization security

Implementing AWS Elemental MediaPackage CDN authorization effectively requires following security best practices for secret management, monitoring, and operational procedures. These recommendations help you maintain secure, cost-effective, and reliable content delivery.

- **Use UUID format for secret values** - We recommend using UUID version 4 format for your secret values, which produces a 36-character string that is both unique and unpredictable.
- **Reuse secrets across endpoints** - When appropriate for your security requirements, use the same secret across multiple endpoints in the same Region and account to reduce management overhead and costs.
- **Implement regular rotation** - Rotate your secrets periodically as part of your security best practices.
- **Monitor authorization failures** - Set up alarms for unusual patterns of authorization failures, which could indicate attempted unauthorized access.
- **Test rotation procedures** - Regularly test your secret rotation procedures to ensure smooth transitions during actual rotations.

Working with data plane APIs in AWS Elemental MediaPackage

MediaPackage uses three data plane actions for ingesting and egressing video content. MediaPackage does not use these actions to configure Channel Group, Channel, or Origin Endpoint resources.

PutObject

During ingest, uploads video segments from an encoder into a MediaPackage channel.

To call PutObject, perform an HTTP GET request to the ingest URL for a channel.

For more information about video format requirements for PutObject, see [Supported input types](#).

GetObject

Download video segments from an origin endpoint in MediaPackage.

To call GetObject, perform an HTTP GET request to an egress URL on an origin endpoint.

GetHeadObject

Retrieves just the HTTP headers of the video content.

To call GetHeadObject, perform an HTTP HEAD request to an egress URL on an origin endpoint.

You can dynamically produce client manifests by appending parameters to the playback request. The following topics provide information about the available query parameters:

- To use query parameters to filter manifests based on audio, subtitle, and video values, see [Manifest filtering from AWS Elemental MediaPackage](#).
- To use query parameters to make live streams available for on-demand viewing for up to 14 days, see [Time-shifted query parameters in AWS Elemental MediaPackage](#).

HarvestObject

Uploads the specified manifest or segment directly to a bucket in Amazon S3. For information about harvesting live-to-VOD assets in MediaPackage, see [Creating live-to-VOD assets with MediaPackage](#).

To call HarvestObject, perform an HTTP POST request to an egress URL, including the manifest or segment path. In the request headers, include the S3 destination bucket name and key.

Example

```
POST egress_domain/channelGroup/ChannelGroupName/channel/ChannelName/  
originEndpoint/OriginEndpointName/manifest_or_segment_path HTTP/1.1
```

```
X-Amz-Source-MediaPackage-S3-Dst-Bucket-Name: amzn-s3-demo-bucket/path
```

```
X-Amz-Source-MediaPackage-S3-Dst-Object-Key: amzn-s3-demo-bucket-key
```

Security in AWS Elemental MediaPackage

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that's built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS compliance programs](#). To learn about the compliance programs that apply to AWS Elemental MediaPackage, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using MediaPackage. The following topics show you how to configure MediaPackage to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your MediaPackage resources.

Topics

- [Data protection in AWS Elemental MediaPackage](#)
- [Identity and Access Management for AWS Elemental MediaPackage](#)
- [Compliance validation for AWS Elemental MediaPackage](#)
- [Resilience in AWS Elemental MediaPackage](#)
- [Infrastructure Security in AWS Elemental MediaPackage](#)

Data protection in AWS Elemental MediaPackage

The AWS [shared responsibility model](#) applies to data protection in AWS Elemental MediaPackage. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted

on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see [Data Privacy FAQ](#). For information about data protection in Europe, see the [General Data Protection Regulation \(GDPR\) Center](#).

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see [Working with CloudTrail trails](#) in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-3](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with MediaPackage or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Topics

- [Implementing DRM with AWS Elemental MediaPackage](#)

Implementing DRM with AWS Elemental MediaPackage

Use encryption to protect your content from unauthorized access. MediaPackage supports digital rights management (DRM). With DRM, you can make sure that once you distribute your content, only authorized viewers can watch it.

For information about using DRM with MediaPackage, see [Content encryption and DRM in AWS Elemental MediaPackage](#).

Identity and Access Management for AWS Elemental MediaPackage

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use MediaPackage resources. IAM is an AWS service that you can use with no additional charge.

Topics

- [Audience](#)
- [Authenticating with identities](#)
- [Managing access using policies](#)
- [How AWS Elemental MediaPackage works with IAM](#)
- [Identity-based policy examples for MediaPackage](#)
- [Resource-based policy examples](#)
- [AWS managed policies for AWS Elemental MediaPackage](#)
- [Authenticating Requests \(AWS Signature Version 4\)](#)
- [Cross-service confused deputy prevention](#)
- [Troubleshooting MediaPackage identity and access](#)
- [Learn More](#)

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see [Troubleshooting MediaPackage identity and access](#))
- **Service administrator** - determine user access and submit permission requests (see [How AWS Elemental MediaPackage works with IAM](#))
- **IAM administrator** - write policies to manage access (see [Identity-based policy examples for MediaPackage](#))

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see [How to sign in to your AWS account](#) in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see [AWS Signature Version 4 for API requests](#) in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see [Tasks that require root user credentials](#) in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see [What is IAM Identity Center?](#) in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An [IAM user](#) is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see [Require human users to use federation with an identity provider to access AWS using temporary credentials](#) in the *IAM User Guide*.

An [IAM group](#) specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see [Use cases for IAM users](#) in the *IAM User Guide*.

IAM roles

An [IAM role](#) is an identity with specific permissions that provides temporary credentials. You can assume a role by [switching from a user to an IAM role \(console\)](#) or by calling an AWS CLI or AWS API operation. For more information, see [Methods to assume a role](#) in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see [Overview of JSON policies](#) in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Define custom IAM permissions with customer managed policies](#) in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see [Choose between managed policies and inline policies](#) in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must [specify a principal](#) in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see [Service control policies](#) in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see [Resource control policies \(RCPs\)](#) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see [Session policies](#) in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy evaluation logic](#) in the *IAM User Guide*.

How AWS Elemental MediaPackage works with IAM

Before you use IAM to manage access to MediaPackage, learn what IAM features are available to use with MediaPackage.

IAM features you can use with MediaPackage

IAM feature	MediaPackage support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Principal permissions	Yes
Service roles	Yes
Service-linked roles	No

To get a high-level view of how MediaPackage and other AWS services work with most IAM features, see [AWS services that work with IAM](#) in the *IAM User Guide*.

Identity-based policies for MediaPackage

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can

perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Define custom IAM permissions with customer managed policies](#) in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see [IAM JSON policy elements reference](#) in the *IAM User Guide*.

Identity-based policy examples for MediaPackage

To view examples of MediaPackage identity-based policies, see [Identity-based policy examples for MediaPackage](#).

Resource-based policies within MediaPackage

Supports resource-based policies: Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

Policy actions for MediaPackage

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The **Action** element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of MediaPackage actions, see [Actions defined by AWS Elemental MediaPackage](#) in the *Service Authorization Reference*.

Policy actions in MediaPackage use the following prefix before the action:

```
mediapackagev2
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [  
    "mediapackagev2:action1",  
    "mediapackagev2:action2"  
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "mediapackagev2:Describe*"
```

To view examples of MediaPackage identity-based policies, see [Identity-based policy examples for MediaPackage](#).

Policy resources for MediaPackage

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its [Amazon Resource Name \(ARN\)](#). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

MediaPackage has the following resource ARNs:

```
arn:${Partition}:mediapackagev2:${Region}:${Account}:channelGroup/${channelGroupName}
arn:${Partition}:mediapackagev2:${Region}:${Account}:channelGroup/${channelGroupName}/
channel/${channelName}
arn:${Partition}:mediapackagev2:${Region}:${Account}:channelGroup/${channelGroupName}/
channel/${channelName}/originEndpoint/${endpointName}
```

For more information about the format of ARNs, see [Amazon Resource Names \(ARNs\) and AWS Service Namespaces](#).

For example, to specify the 9a6b3953e242400eb805f324d95788e3 channel in your statement, use the following ARN:

```
"Resource": "arn:aws:mediapackagev2:us-
east-1:111122223333:channelGroup/channelGroupName/
channel/9a6b3953e242400eb805f324d95788e3"
```

To specify all instances that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:mediapackagev2:us-
east-1:111122223333:channelGroup/channelGroupName/channel/*"
```

Some MediaPackage actions, such as those for creating resources, can't be performed on a specific resource. In those cases, you must use the wildcard (*).

```
"Resource": "*"
```

To see a list of MediaPackage resource types and their ARNs, see [Resources defined by AWS Elemental MediaPackage](#) in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see [Actions defined by AWS Elemental MediaPackage](#).

To view examples of MediaPackage identity-based policies, see [Identity-based policy examples for MediaPackage](#).

Policy condition keys for MediaPackage

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use [condition operators](#), such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see [AWS global condition context keys](#) in the *IAM User Guide*.

To see a list of MediaPackage condition keys, see [Condition keys for AWS Elemental MediaPackage](#) in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see [Actions defined by AWS Elemental MediaPackage](#).

The `mediapackagev2:RequestHasMatchingCdnAuthHeader` condition key is a Boolean type key that you can use to control access based on whether a request has a matching CDN authentication header. This condition key is useful when implementing CDN authorization as described in [Secure MediaPackage content with CDN authorization](#). You can use this condition key in your IAM policies to allow or deny actions based on the presence of a valid CDN authentication header in the request.

For example, you can create a policy that allows access to content only when the request includes a valid CDN authentication header:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGetObjectAccessForAuthorizedRequest",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "mediapackagev2:GetObject",
      "Resource": "arn:aws:mediapackagev2:us-east-1:111122223333:channelGroup/channelGroupName/channel/channelName/originEndpoint/originEndpointName",
      "Condition": {
        "Bool": {
          "mediapackagev2:RequestHasMatchingCdnAuthHeader": "true"
        }
      }
    }
  ]
}
```

```
]
}
```

To view examples of MediaPackage identity-based policies, see [Identity-based policy examples for MediaPackage](#).

ACLs in MediaPackage

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with MediaPackage

Supports ABAC (tags in policies): Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the [condition element](#) of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see [Define permissions with ABAC authorization](#) in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see [Use attribute-based access control \(ABAC\)](#) in the *IAM User Guide*.

Using temporary credentials with MediaPackage

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate

temporary credentials instead of using long-term access keys. For more information, see [Temporary security credentials in IAM](#) and [AWS services that work with IAM](#) in the *IAM User Guide*.

Cross-service principal permissions for MediaPackage

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see [Forward access sessions](#).

Service roles for MediaPackage

Supports service roles: Yes

A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Create a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

Warning

Changing the permissions for a service role might break MediaPackage functionality. Edit service roles only when MediaPackage provides guidance to do so.

Choosing an IAM role in MediaPackage

When you create an asset resource in MediaPackage, you must choose a role to allow MediaPackage to access Amazon S3 on your behalf. If you previously created a service role or service-linked role, MediaPackage provides you with a list of roles to choose from. It's important to choose a role that allows access to read from the Amazon S3 bucket and retrieve content.

Service-linked roles for MediaPackage

Supports service-linked roles: No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see [AWS services that work with IAM](#). Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for MediaPackage

By default, users and roles don't have permission to create or modify MediaPackage resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see [Create IAM policies \(console\)](#) in the *IAM User Guide*.

For details about actions and resource types defined by MediaPackage, including the format of the ARNs for each of the resource types, see [Actions, resources, and condition keys for AWS Elemental MediaPackage](#) in the *Service Authorization Reference*.

Topics

- [Policy best practices](#)
- [Using the MediaPackage console](#)
- [Allow users to view their own permissions](#)

Policy best practices

Identity-based policies determine whether someone can create, access, or delete MediaPackage resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see [AWS managed policies](#) or [AWS managed policies for job functions](#) in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more

information about using IAM to apply permissions, see [Policies and permissions in IAM](#) in the *IAM User Guide*.

- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see [Validate policies with IAM Access Analyzer](#) in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see [Secure API access with MFA](#) in the *IAM User Guide*.

For more information about best practices in IAM, see [Security best practices in IAM](#) in the *IAM User Guide*.

Using the MediaPackage console

To access the AWS Elemental MediaPackage console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the MediaPackage resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the MediaPackage console, also attach the MediaPackage *ReadOnly* AWS managed policy to the entities. For more information, see [Adding permissions to a user](#) in the *IAM User Guide*.

AWS Elemental MediaPackageReadOnly

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

Resource-based policy examples

A resource policy is an access policy option available for granting permission to your MediaPackage resources. [Resource-based policies](#) are JSON policy documents.

The topics in this section describe the key policy language elements, with focus on MediaPackage-specific details, and provide example resource policies. We recommend that you first review the introductory topics that explain the basic concepts and options available for you to manage access to your MediaPackage resources.

To learn how to attach a resource-based policy to a channel, see [Creating a channel in AWS Elemental MediaPackage](#).

Topics

- [Policies and Permissions in MediaPackage](#)
- [Ingest authorization](#)
- [Origin endpoint authorization](#)

Policies and Permissions in MediaPackage

This page provides an overview of resource policies in MediaPackage and describes the basic elements of a policy. Each listed element links to more details about that element and examples of how to use it.

For a complete list of MediaPackage actions, resources, and conditions, see [Actions, resources, and condition keys for AWS Elemental MediaPackage](#) in the *AWS General Reference*.

In its most basic sense, a policy contains the following elements:

- **Resources** - Channels and origin endpoints are the MediaPackage resources for which you can allow or deny permissions. In a policy, you use the Amazon Resource Name (ARN) to identify the resource. For more information, see [MediaPackage resources](#).

Important

Wildcards are not allowed in the resource ARN in [resource-based policies](#). The policy must contain the explicit ARN for each resource that it applies to.

- **Actions** - For each resource, MediaPackage supports a set of operations. You identify resource operations that you will allow (or deny) by using action keywords. For more information, see [IAM JSON Policy Elements: Action](#).
- **Effect** - This determines what the effect will be when the user requests the specific action. This can be either *allow* or *deny*.

If you do not explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource. You might do this to make sure that a user can't access the resource, even if a different policy grants access. For more information, see [IAM JSON Policy Elements: Effect](#).

- **Principal** - The account or user who is allowed access to the actions and resources in the statement. In a resource policy, the principal is the user, account, service, or other entity that is the recipient of this permission. For more information, see [Principals](#) and [AWS JSON Policy Elements: Principal](#).
- **Condition** - These are the conditions for when a policy is in effect. You can use AWS-wide keys and MediaPackage-specific keys to specify conditions in an MediaPackage access policy. For more information, see [IAM JSON Policy Elements: Condition](#).

To illustrate, consider the following Allow policy. With this policy in effect, Jane Doe has `mediapackagev2:GetObject` and `mediapackagev2:GetHeadObject` permissions on all objects from the specified origin endpoint under the condition that the request are made over HTTPS.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowJaneDoe",
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:user/JaneDoe" },
      "Action": ["mediapackagev2:GetObject", "mediapackagev2:GetHeadObject"],
      "Resource": "arn:aws:mediapackagev2:us-east-1:111122223333:channelGroup/ChannelGroupName/channel/ChannelName/originEndpoint/OriginEndpointName",
      "Condition": {
        "Bool": { "aws:SecureTransport": "true" }
      }
    }
  ]
}
```

```
}  
}  
]  
}
```

Resource policies are specific to the resources to which they are applied. You must apply the policy explicitly to each resource that requires it.

For example, applying a policy to a particular origin endpoint that allows anonymous `GetObject` doesn't automatically apply `GetObject` to other endpoints even if the ARN matches. For instance, if you apply a policy to origin endpoint `abcdef01234567890`, it only applies to that endpoint and not to another endpoint with a similar ARN, like `021345abcdef6789`.

For more, see the topics below. For complete policy language information, see [Policies and Permissions](#) and [IAM JSON Policy Reference](#) in the *IAM User Guide*.

Topics

- [Principals](#)
- [Actions, resources, and condition keys in MediaPackage](#)

Principals

The `Principal` element specifies the user, account, service, or other entity that is allowed or denied access to a resource. For more information, see [Principal](#) in the *IAM User Guide*.

Grant permissions to an AWS account

To grant permissions to an AWS account, identify the account using the following format.

```
"AWS": "account-ARN"
```

The following are examples.

```
"Principal": {"AWS": "arn:aws:iam::AccountIDWithoutHyphens:root"}
```

```
"Principal": {"AWS":  
  ["arn:aws:iam::AccountID1WithoutHyphens:root", "arn:aws:iam::AccountID2WithoutHyphens:root"]}]
```

Grant permissions to an IAM user

To grant permission to an IAM user within your account, you must provide an "AWS" : "*user-ARN*" name-value pair.

```
"Principal":{"AWS":"arn:aws:iam::account-number-without-hyphens:user/username"}
```

Note

If an IAM identity is deleted after you update your resource policy, the resource policy will show a unique identifier in the principal element instead of an ARN. These unique IDs are never reused, so you can safely remove principals with unique identifiers from all of your policy statements. For more information about unique identifiers, see [IAM identifiers](#) in the *IAM User Guide*.

Grant anonymous permissions

To grant permission to everyone, also referred as anonymous access, you set the wildcard ("*") as the Principal value. For example, if you want to use clients with no AWS authorization to their origin endpoints.

```
"Principal": "*"
```

```
"Principal":{"AWS":"*"}
```

Using "Principal": "*" with an Allow effect in a resource-based policy allows anyone, even if they're not signed in to AWS, to access your resource.

Using "Principal" : { "AWS" : "*" } with an Allow effect in a resource-based policy allows any root user, IAM user, assumed-role session, or federated user in any account in the same partition to access your resource.

For anonymous users, these two methods are equivalent. For more information, see [All principals](#) in the *IAM User Guide*.

You cannot use a wildcard to match part of a principal name or ARN.

⚠ Important

Because anyone can create an AWS account, the **security level** of these two methods is equivalent, even though they function differently.

⚠ Warning

Use caution when granting anonymous access to your MediaPackage origin endpoints. When you grant anonymous access, anyone in the world can access your bucket. We highly recommend that you never grant any kind of anonymous write access to your origin endpoints.

Actions, resources, and condition keys in MediaPackage

AWS Elemental MediaPackage (service prefix: `mediapackagev2`) provides service-specific resources, actions, and condition context keys for use in IAM permission policies. For the full list, see [Actions, resources, and condition keys for AWS Elemental MediaPackage](#) in the *AWS General Reference*.

MediaPackage Actions

MediaPackage defines a set of permissions that you can specify in a policy. These are keywords, each of which maps to a specific MediaPackage operation. When you use an action in a policy, you usually allow or deny access to the API operation or CLI command with the same name. However, in some cases, a single action controls access to more than one operation. Alternatively, some operations require several different actions.

MediaPackage resources

The following common Amazon Resource Name (ARN) format identifies resources in AWS:

```
arn:${Partition}:mediapackagev2:${Region}:${AccountID}:channelGroup/  
${ChannelGroupName}/channel/${ChannelName}/originEndpoint/${OriginEndpointName}
```

For information about ARNs, see [Amazon Resource Names \(ARNs\)](#) in the *AWS General Reference*.

For information about resources, see [IAM JSON Policy Elements: Resource](#) in the *IAM User Guide*.

A MediaPackage ARN includes the following:

- **Partition** - aws is a common partition name. If your resources are in the China (Beijing) Region, aws-cn is the partition name.
- **Region** - The AWS Region.
- **AccountID** - Your AWS account number.
- **ChannelGroupName** - The name of the channel group.
- **ChannelName** - The name of the channel.
- **OriginEndpointName** - The name of the origin endpoint.

MediaPackage Conditions keys

The access policy language enables you to specify conditions when granting permissions. To specify conditions for when a policy is in effect, you can use the optional Condition element, or Condition block, to specify conditions for when a policy is in effect. You can use predefined AWS-wide keys and MediaPackage-specific keys to specify conditions in an MediaPackage access policy. In the Condition element, you build expressions in which you use Boolean operators (equal, less than, etc.) to match your condition against values in the request.

Ingest authorization

MediaPackage ingest requests usually originate from a video encoder.

Topics

- [AWS Elemental MediaLive](#)
- [AWS Elemental Live](#)
- [Third-party encoders](#)
- [Clients that don't support AWS authorization](#)

AWS Elemental MediaLive

This example illustrates a channel policy that permits MediaLive to ingest MediaPackage.

JSON

```
{
```

```

"Version": "2012-10-17",
"Id": "AllowMediaLiveChannelToIngestToEmpChannel",
"Statement": [
  {
    "Sid": "AllowMediaLiveRoleToAccessEmpChannel",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111122223333:role/MediaLiveAccessRole"
    },
    "Action": "mediapackagev2:PutObject",
    "Resource": "arn:aws:mediapackagev2:us-east-1:111122223333:channelGroup/ChannelGroupName/channel/ChannelName"
  }
]
}

```

AWS Elemental Live

If you provide Elemental Live with an access key ID and secret access key, it can request access as an IAM identity. To grant your Elemental Live encoder access to your MediaPackage channel, you can apply the following Allow policy.

1. In IAM, create an IAM user such as `ElementalLiveMediaPackageUser` with **Programmatic access**.
2. In MediaPackage, create or edit a channel to include the following channel policy.

JSON

```

{
  "Version": "2012-10-17",
  "Id": "AllowIamUser",
  "Statement": [
    {
      "Sid": "AllowIamUserToEmpChannel",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:user/ElementalLiveMediaPackageUser"
      },
      "Action": "mediapackagev2:PutObject",
      "Resource": "arn:aws:mediapackagev2:us-east-1:111122223333:channelGroup/ChannelGroupName/channel/ChannelName"
    }
  ]
}

```

```
}  
]  
}
```

3. In IAM, create an access key for `ElementalLiveMediaPackageAccessUser`. Save the access key .csv file in a secure location to retain a permanent record of the access key ID and secret access key.

The access key ID looks like this: AKIAIOSFODNN7EXAMPLE

The secret access key looks like this: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

For more information, see [Programmatic access](#) in the *AWS General Reference*.

4. Share the access key ID and the secret access key with the Elemental Live operator. Do *not* give the username and password to the operator.

By following these steps, you'll create an AWS user with the necessary permissions required to allow Elemental Live to make requests to MediaPackage. When the operator sets up the output with MediaPackage as the destination, they will enter the access key ID and secret access key. During the Elemental Live event, Elemental Live sends these two IDs to the AWS service instead of the username and password, providing authorization to AWS for the Elemental Live node to make requests to MediaPackage.

Third-party encoders

Third-party encoders that support AWS authorization operate similarly to Elemental Live, as described earlier. To grant access, create an IAM user and a MediaPackage channel resource policy that permits the user to call `PutObject`. On the encoder's side, use the IAM user access key ID and secret access key to sign the requests.

Clients that don't support AWS authorization

Encoders that don't support AWS authorization can be granted ingest access by restricting access to specific IP ranges using the `aws:SourceIp` condition key. This is useful for third-party encoders that can't sign requests with IAM credentials. For information about condition keys, see [IAM JSON Policy Elements: Condition](#).

MediaPackage doesn't support anonymous access for `PutObject` API calls. You must scope a wildcard principal (`"Principal": "*"`) with an `aws:SourceIp` condition.

Restrict access by IP range

Consider the following Allow policy. With this policy in effect, MediaPackage restricts ingest to IP addresses in the range 203.0.113.0 to 203.0.113.255 using the `aws:SourceIp` condition key.

```
{
  "Version": "2012-10-17",
  "Id": "IpRangePolicy",
  "Statement": [
    {
      "Sid": "RestrictByIpRange",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "mediapackagev2:PutObject",
      "Resource": "arn:aws:mediapackagev2:Region:AccountID:channelGroup/ChannelGroupName/channel/ChannelName",
      "Condition": {
        "IpAddress": { "aws:SourceIp": "203.0.113.0/24" }
      }
    }
  ]
}
```

Origin endpoint authorization

MediaPackage egress requests usually originate from CDNs, but they may also come from other sources such as customer-owned monitoring scripts or operators using web browsers like Safari or Chrome to view the video stream and identify any issues.

Topics

- [MediaPackage L2V Harvester](#)
- [Third-party CDNs that support AWS authorization](#)
- [Clients that don't support AWS authorization](#)

MediaPackage L2V Harvester

To allow MediaPackage harvest jobs to get content from your origin endpoint, create or edit an origin endpoint with the following endpoint policy. For more information about harvest jobs, see [Creating live-to-VOD assets with MediaPackage](#).

JSON

```
{
  "Version": "2012-10-17",
  "Id": "MediaPackageHarvesterAccessPolicy",
  "Statement": [
    {
      "Sid": "AllowMediaPackageHarvestObjectAccess",
      "Effect": "Allow",
      "Principal": {
        "Service": "mediapackagev2.amazonaws.com"
      },
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "111122223333"
        }
      },
      "Action": [
        "mediapackagev2:HarvestObject",
        "mediapackagev2:GetObject"
      ],
      "Resource": "arn:aws:mediapackagev2:us-  
east-1:111122223333:channelGroup/ChannelGroupName/channel/ChannelName/  
originEndpoint/OriginEndpointName"
    }
  ]
}
```

Third-party CDNs that support AWS authorization

To authorize an external CDN that supports AWS authorization, you need to create a specific IAM user for the CDN, allow access in their origin endpoint policy, and provide the CDN with the AWS access key ID and secret access key for the IAM user. For example, if you want to give your CDN provider access to your MediaPackage origin endpoint, you can follow the following procedure.

1. In IAM, create an IAM user such as `CDNProviderMediaPackageAccessUser` with **Programmatic access**.
2. In MediaPackage, create or edit an origin endpoint to include the following endpoint policy.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "PolicyForCDNProviderPrivateContent",
  "Statement": [
    {
      "Sid": "AllowCDNProviderUser",
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::111122223333:user/CDNProviderMediaPackageAccessUser" },
      "Action": "mediapackagev2:GetObject",
      "Resource": "arn:aws:mediapackagev2:us-east-1:111122223333:channelGroup/ChannelGroupName/channel/ChannelName/originEndpoint/OriginEndpointName"
    }
  ]
}
```

3. In IAM, create an access key for `CDNProviderMediaPackageAccessUser`. Save the access key .csv file in a secure location to retain a permanent record of the access key ID and secret access key.

The access key ID looks like this: AKIAIOSFODNN7EXAMPLE

The secret access key looks like this: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

For more information, see [Programmatic access](#) in the *AWS General Reference*.

4. Follow the instructions in your CDN provider's documentation for authenticating with AWS access keys.

By following these steps, you'll create an AWS user with the necessary permissions required to allow the external CDN make requests to MediaPackage. When the CDN provider sets up the output with MediaPackage as the destination, they will enter the access key ID and secret access key. During the event, the provider sends these two IDs to the AWS service instead of the username and password, providing authorization to make requests to MediaPackage.

Clients that don't support AWS authorization

Clients without AWS authorization support can be granted access to origin endpoints either by enabling anonymous access or by restricting access to specific IP ranges using the `aws:SourceIp` condition key. This is useful for clients such as external CDNs that don't support AWS authorization, as well as monitoring scripts and human operators who may use web browsers to visually inspect a video stream. For information about condition keys, see [IAM JSON Policy Elements: Condition](#).

Anonymous access

Consider the following Allow policy. With this policy in effect, MediaPackage allows anonymous access to the `mediapackagev2:GetObject` action on the channel resource in the policy.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "AnonymousAccessPolicy",
  "Statement": [
    {
      "Sid": "AllowAnonymousAccess",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "mediapackagev2:GetObject",
      "Resource": "arn:aws:mediapackagev2:us-  
east-1:111122223333:channelGroup/ChannelGroupName/channel/ChannelName/  
originEndpoint/OriginEndpointName"
    }
  ]
}
```

MediaPackage doesn't support anonymous access for `PutObject` API calls.

Cross-account access

Consider the following Allow policy. With this policy in effect, MediaPackage allows, across accounts (`accountID` and `differentAccountID`), the `mediapackagev2:GetObject` action on the channel resource in the policy.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCrossAccountAccess",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::444455556666:root"},
      "Action": "mediapackagev2:GetObject",
      "Resource": "arn:aws:mediapackagev2:us-east-1:111122223333:channelGroup/ChannelGroupName/channel/ChannelName"
    }
  ]
}
```

Restrict access by IP range

Consider the following Allow policy. With this policy in effect, MediaPackage restricts access to IP addresses in the range 203.0.113.0 to 203.0.113.255 using the `aws:SourceIp` condition key. For information about condition keys, see [IAM JSON Policy Elements: Condition](#).

JSON

```
{
  "Version": "2012-10-17",
  "Id": "IpRangePolicy",
  "Statement": [
    {
      "Sid": "RestrictByIpRange",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "mediapackagev2:GetObject",
      "Resource": "arn:aws:mediapackagev2:us-east-1:111122223333:channelGroup/ChannelGroupName/channel/ChannelName/originEndpoint/OriginEndpointName",
      "Condition": {
        "IpAddress": { "aws:SourceIp": "203.0.113.0/24" }
      }
    }
  ]
}
```

```
]
}
```

AWS managed policies for AWS Elemental MediaPackage

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining [customer managed policies](#) that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see [AWS managed policies](#) in the *IAM User Guide*.

AWS managed policy: AWSElementalMediaPackageV2FullAccess

This policy grants contributor permissions that allow all actions on all live resources in MediaPackage.

You can attach the AWSElementalMediaPackageV2FullAccess policy to your IAM identities.

To view the permissions for this policy, see [AWSElementalMediaPackageV2FullAccess](#) in the *AWS Managed Policy Reference*.

AWS managed policy: AWSElementalMediaPackageV2ReadOnly

This policy grants contributor permissions that allow read-only actions on all live resources in MediaPackage.

You can attach the AWSElementalMediaPackageV2ReadOnly policy to your IAM identities.

To view the permissions for this policy, see [AWSElementalMediaPackageV2ReadOnly](#) in the *AWS Managed Policy Reference*.

MediaPackage updates to AWS managed policies

View details about updates to AWS managed policies for MediaPackage since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the MediaPackage [Document history](#) page.

Change	Description	Date
AWSElementalMediaPackageV2FullAccess – New policy	MediaPackage added a new full-access policy for live resources. This policy allows all actions on all live resources in MediaPackage.	July 25, 2023
AWSElementalMediaPackageV2ReadOnly – New policy	MediaPackage added a new read-only policy for live resources.	July 25, 2023

Change	Description	Date
	This policy allows read-only actions on all live resources in MediaPackage.	
MediaPackage started tracking changes	MediaPackage started tracking changes for its AWS managed policies.	July 25, 2023

Authenticating Requests (AWS Signature Version 4)

Every interaction with MediaPackage is either authenticated or anonymous. This section explains request authentication with the AWS Signature Version 4 algorithm.

Note

If you use the AWS SDKs or AWS CLI to send your requests, you don't need to read this section because these tools authenticate your requests by using access keys that you provide. You must only sign AWS API requests as described in this documentation if you do not use an AWS SDK or AWS CLI to send AWS API requests.

When you send API requests to AWS, you must sign them so that AWS can identify the sender. For security, most requests are signed using your AWS security credentials.

When MediaPackage receives an authenticated request, it recreates the signature using the authentication information contained in the request. If the signatures match, MediaPackage processes the request. Otherwise, it rejects the request.

AWS Signature Version 4 is the AWS signing protocol. AWS also supports an extension, Signature Version 4A, which supports signatures for multi-Region API requests.

For additional information about AWS Signature Version 4, see:

- [Signing AWS API requests](#) in the *IAM User Guide*
- [Authenticating Requests \(AWS Signature Version 4\)](#) in the *Amazon Simple Storage Service API Reference*

Creating a signed AWS API request

For steps to create a signed AWS API request, see:

- [Create a signed AWS API request](#) in the *IAM User Guide*
- [Signature Calculations for the Authorization Headers: Transferring Payload in a Single Chunk](#) in the *Amazon Simple Storage Service API Reference*

Troubleshooting signed AWS API requests

For troubleshooting help with your signed requests, see [Troubleshoot signed requests for AWS APIs](#) in the *IAM User Guide*.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the [aws:SourceArn](#) and [aws:SourceAccount](#) global condition context keys in resource policies to limit the permissions that AWS Elemental MediaPackage gives another service to the resource. Use `aws:SourceArn` if you want only one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the resource or if you are specifying multiple resources, use the `aws:SourceArn` global context condition key with wildcard characters (*) for the unknown portions of the ARN. For example, `arn:aws:servicename:*:123456789012:*`.

If the `aws:SourceArn` value does not contain the account ID, such as an Amazon S3 bucket ARN, you must use both global condition context keys to limit permissions.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in MediaPackage to prevent the confused deputy problem.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "mediapackage.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:mediapackage:*:123456789012:harvest_jobs/*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Troubleshooting MediaPackage identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with MediaPackage and IAM.

Topics

- [I'm not authorized to perform an action in MediaPackage](#)
- [I'm not authorized to perform iam:PassRole](#)
- [I want to allow people outside of my AWS account to access my MediaPackage resources](#)

I'm not authorized to perform an action in MediaPackage

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *my-example-widget* resource but doesn't have the fictional `mediapackagev2:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
mediapackagev2:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *my-example-widget* resource by using the `mediapackagev2:GetWidget` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I'm not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to MediaPackage.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in MediaPackage. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my MediaPackage resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether MediaPackage supports these features, see [How AWS Elemental MediaPackage works with IAM](#).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing access to an IAM user in another AWS account that you own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing access to AWS accounts owned by third parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing access to externally authenticated users \(identity federation\)](#) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

Learn More

For more information about identity and access management for MediaPackage, continue to the following pages:

- [How AWS Elemental MediaPackage works with IAM](#)
- [Identity-based policy examples for MediaPackage](#)
- [Cross-service confused deputy prevention](#)
- [Troubleshooting MediaPackage identity and access](#)

Compliance validation for AWS Elemental MediaPackage

Our new AWS sign-up experience is not designed for regulated workloads. If you're using our new AWS sign-up experience, but you want to use AWS for regulated workloads, you can [sign up for AWS \(advanced\)](#) or [activate advanced features](#) for your AWS environment.

To learn whether an AWS service is within the scope of specific compliance programs, see [AWS services in Scope by Compliance Program](#) and choose the compliance program that you are interested in. For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see [AWS Security Documentation](#).

Resilience in AWS Elemental MediaPackage

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS global infrastructure](#).

Infrastructure Security in AWS Elemental MediaPackage

As a managed service, AWS Elemental MediaPackage is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see [AWS Cloud Security](#). To design your AWS environment using the best practices for infrastructure security, see [Infrastructure Protection](#) in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access MediaPackage through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Logging and monitoring in MediaPackage

Monitoring is an important part of maintaining the reliability, availability, and performance of MediaPackage and your other AWS solutions. AWS provides the following monitoring tools to watch MediaPackage, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and the applications that you run on AWS in real-time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the [Amazon CloudWatch User Guide](#).
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the [AWS CloudTrail User Guide](#).

Topics

- [Monitoring AWS Elemental MediaPackage with Amazon CloudWatch metrics](#)
- [Monitoring AWS Elemental MediaPackage with EventBridge events](#)
- [Logging AWS Elemental MediaPackage API calls with AWS CloudTrail](#)
- [Access logging](#)
- [Monitoring manifest update time in AWS Elemental MediaPackage](#)
- [MediaPackage response headers](#)

Monitoring AWS Elemental MediaPackage with Amazon CloudWatch metrics

You can monitor MediaPackage using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or

take actions when those thresholds are met. For more information, see the [Amazon CloudWatch User Guide](#).

MediaPackage console

MediaPackage displays metrics throughout the console.

To view metrics using the MediaPackage console

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.
2. Navigate to the appropriate page to view metrics:
 - For metrics on all channel groups, go to the **Channel groups** page.
 - For metrics on all channels and origin endpoints associated with your channel group in the AWS Region, go to the channel group's details page.
 - For metrics on a specific channel and all of its origin endpoints, go to the channel's details page.
 - For metrics on a specific origin endpoint, go to the origin endpoint's details page.
3. (Optional) To refine the metrics view, choose **Open in CloudWatch**.

CloudWatch console

Metrics are grouped first by the service namespace, and then by the various dimension combinations within each namespace.

To view metrics using the CloudWatch console

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Under **All metrics**, choose the **AWS/MediaPackage** namespace.
4. Choose the metric dimension to view the metrics (for example, choose `channel` to view metrics per channel).

AWS CLI

To view metrics using the AWS CLI

At a command prompt, enter the following command:

- ```
aws cloudwatch list-metrics --namespace "AWS/MediaPackage"
```

## MediaPackage live content metrics

The AWS/MediaPackage namespace includes the following metrics for live content. MediaPackage publishes metrics to CloudWatch every minute, if not sooner.

| Metric      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelMQCS | <p>Segment-level quality score as calculated by MediaPackage for the active input to this channel.</p> <p>Units: Numeric</p> <p>Valid statistics:</p> <ul style="list-style-type: none"><li>Average – Average bytes (Sum/SampleCount ) that MediaPackage outputs over the configured interval.</li><li>Maximum – Largest individual output request (in bytes) made to MediaPackage.</li><li>Minimum – Smallest individual output request (in bytes) made to MediaPackage.</li><li>SampleCount – Number of requests that's used in the statistical calculation.</li><li>Sum – Total number of bytes that MediaPackage outputs over the configured interval.</li></ul> <p>Valid dimensions:</p> |

| Metric              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>• ChannelGroup</li> <li>• Combination of ChannelGroup and Channel</li> <li>• Combination of ChannelGroup , Channel, and TrackType</li> <li>• No dimension</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| ChannelMQCSSequence | <p>Aggregated quality score for all segments in the sequence for the active input to this channel, as calculated by MediaPackage.</p> <p>Units: Numeric</p> <p>Valid statistics:</p> <ul style="list-style-type: none"> <li>• Average – Average bytes (Sum/SampleCount ) that MediaPackage outputs over the configured interval.</li> <li>• Maximum – Largest individual output request (in bytes) made to MediaPackage.</li> <li>• Minimum – Smallest individual output request (in bytes) made to MediaPackage.</li> <li>• SampleCount – Number of requests that's used in the statistical calculation.</li> <li>• Sum – Total number of bytes that MediaPackage outputs over the configured interval.</li> </ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"> <li>• ChannelGroup</li> <li>• Combination of ChannelGroup and Channel</li> <li>• No dimension</li> </ul> |

| Metric      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EgressBytes | <p>Number of bytes that MediaPackage successfully sends for each request. If MediaPackage doesn't receive any requests for output in the specified interval, then no data is given.</p> <p>Units: Bytes</p> <p>Valid statistics:</p> <ul style="list-style-type: none"> <li>• <b>Average</b> – Average bytes (Sum/SampleCount) that MediaPackage outputs over the configured interval.</li> <li>• <b>Maximum</b> – Largest individual output request (in bytes) made to MediaPackage.</li> <li>• <b>Minimum</b> – Smallest individual output request (in bytes) made to MediaPackage.</li> <li>• <b>SampleCount</b> – Number of requests that's used in the statistical calculation.</li> <li>• <b>Sum</b> – Total number of bytes that MediaPackage outputs over the configured interval.</li> </ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"> <li>• ChannelGroup</li> <li>• RequestType</li> <li>• Combination of ChannelGroup and Channel</li> <li>• Combination of ChannelGroup, Channel, and RequestType</li> <li>• Combination of ChannelGroup, Channel, and OriginEndpoint</li> <li>• Combination of ChannelGroup, Channel, OriginEndpoint, and RequestType</li> </ul> |

| Metric | Description                                                                                                                                                                                  |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | <ul style="list-style-type: none"><li>• Combination of ChannelGroup , Channel, OriginEndpoint , RequestType , and MultiviewLayout</li><li>• MultiviewLayout</li><li>• No dimension</li></ul> |

| Metric             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EgressRequestCount | <p>Number of content requests that MediaPackage receives. If MediaPackage doesn't receive any requests for output in the specified interval, then no data is given.</p> <p>Units: Count</p> <p>Valid statistics:</p> <ul style="list-style-type: none"> <li>Sum – Total number of output requests that MediaPackage receives.</li> </ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"> <li>ChannelGroup</li> <li>RequestType</li> <li>StatusCode</li> <li>OutputType and StatusCode</li> <li>Combination of ChannelGroup and RequestType</li> <li>Combination of ChannelGroup and StatusCode</li> <li>Combination of ChannelGroup , RequestType , and StatusCode</li> <li>Combination of ChannelGroup and Channel</li> <li>Combination of ChannelGroup , Channel, and RequestType</li> <li>Combination of ChannelGroup , Channel, and OriginEndpoint</li> <li>Combination of ChannelGroup , Channel, and StatusCode</li> </ul> |

| Metric | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | <ul style="list-style-type: none"><li>• Combination of ChannelGroup , Channel, OriginEndpoint , and RequestType</li><li>• Combination of ChannelGroup , Channel, RequestType , and StatusCode</li><li>• Combination of ChannelGroup , Channel, OriginEndpoint , and StatusCode</li><li>• Combination of ChannelGroup , Channel, OriginEndpoint , RequestType , and StatusCode</li><li>• Combination of ChannelGroup , Channel, OriginEndpoint , RequestType , StatusCode , and MultiviewLayout</li><li>• MultiviewLayout</li><li>• No dimension</li></ul> |

| Metric             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EgressResponseTime | <p>The time that it takes MediaPackage to process each output request. If MediaPackage doesn't receive any requests for output in the specified interval, then no data is given.</p> <p>Units: Milliseconds</p> <p>Valid statistics:</p> <ul style="list-style-type: none"><li>• <b>Average</b> – Average amount of time (Sum/SampleCount ) that it takes MediaPackage to process output requests over the configured interval.</li><li>• <b>Maximum</b> – Longest amount of time (in milliseconds) that it takes MediaPackage to process an output request and provide a response.</li><li>• <b>Minimum</b> – Shortest amount of time (in milliseconds) that it takes MediaPackage to process an output request and provide a response.</li><li>• <b>SampleCount</b> – Number of requests that's used in the statistical calculation.</li><li>• <b>Sum</b> – Total amount of time that it takes MediaPackage to process output requests over the configured interval.</li></ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"><li>• ChannelGroup</li><li>• RequestType</li><li>• Combination of ChannelGroup and Channel</li></ul> |

| Metric | Description                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | <ul style="list-style-type: none"><li>• Combination of ChannelGroup , Channel, and RequestType</li><li>• Combination of ChannelGroup , Channel, and OriginEndpoint</li><li>• Combination of ChannelGroup , Channel, OriginEndpoint , and RequestType</li><li>• Combination of ChannelGroup , Channel, OriginEndpoint , RequestType , and MultiviewLayout</li><li>• MultiviewLayout</li><li>• No dimension</li></ul> |

| Metric       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IngressBytes | <p>Number of bytes of content that MediaPackage receives for each input request. If MediaPackage doesn't receive any requests for input in the specified interval, then no data is given.</p> <p>Units: Bytes</p> <p>Valid statistics:</p> <ul style="list-style-type: none"> <li>• <b>Average</b> – Average bytes (Sum/SampleCount) that MediaPackage receives over the configured interval.</li> <li>• <b>Maximum</b> – Largest individual input request (in bytes) made to MediaPackage.</li> <li>• <b>Minimum</b> – Smallest individual input request (in bytes) made to MediaPackage.</li> <li>• <b>SampleCount</b> – Number of requests that's used in the statistical calculation.</li> <li>• <b>Sum</b> – Total number of bytes that MediaPackage receives over the configured interval.</li> </ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"> <li>• ChannelGroup</li> <li>• Combination of ChannelGroup and Channel</li> <li>• Combination of ChannelGroup, Channel, and IngestEndpoint</li> <li>• No dimension</li> </ul> |

| Metric      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IngressMQCS | <p>Segment-level quality score as communicated by AWS Elemental MediaLive for this input.</p> <p>Units: Numeric</p> <p>Valid statistics:</p> <ul style="list-style-type: none"><li>• <b>Average</b> – Average bytes (Sum/SampleCount ) that MediaPackage outputs over the configured interval.</li><li>• <b>Maximum</b> – Largest individual output request (in bytes) made to MediaPackage.</li><li>• <b>Minimum</b> – Smallest individual output request (in bytes) made to MediaPackage.</li><li>• <b>SampleCount</b> – Number of requests that's used in the statistical calculation.</li><li>• <b>Sum</b> – Total number of bytes that MediaPackage outputs over the configured interval.</li></ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"><li>• ChannelGroup</li><li>• Combination of ChannelGroup and Channel</li><li>• Combination of ChannelGroup , Channel, and IngestEndpoint</li><li>• Combination of ChannelGroup , Channel, IngestEndpoint , and TrackType</li><li>• No dimension</li></ul> |

| Metric             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IngressMQCSequence | <p data-bbox="829 226 1446 359">Aggregated quality scores for all segments in the sequence, as communicated by AWS Elemental MediaLive.</p> <p data-bbox="829 401 1044 436">Units: Numeric</p> <p data-bbox="829 478 1044 514">Valid statistics:</p> <ul data-bbox="829 556 1503 1115" style="list-style-type: none"><li data-bbox="829 556 1503 695">• Average – Average bytes (Sum/SampleCount ) that MediaPackage outputs over the configured interval.</li><li data-bbox="829 716 1463 793">• Maximum – Largest individual output request (in bytes) made to MediaPackage.</li><li data-bbox="829 814 1463 892">• Minimum – Smallest individual output request (in bytes) made to MediaPackage.</li><li data-bbox="829 913 1503 991">• SampleCount – Number of requests that's used in the statistical calculation.</li><li data-bbox="829 1012 1503 1115">• Sum – Total number of bytes that MediaPackage outputs over the configured interval.</li></ul> <p data-bbox="829 1188 1084 1224">Valid dimensions:</p> <ul data-bbox="829 1266 1503 1570" style="list-style-type: none"><li data-bbox="829 1266 1089 1302">• ChannelGroup</li><li data-bbox="829 1323 1406 1400">• Combination of ChannelGroup and Channel</li><li data-bbox="829 1421 1503 1499">• Combination of ChannelGroup , Channel, and IngestEndpoint</li><li data-bbox="829 1520 1062 1556">• No dimension</li></ul> |

| Metric              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IngressRequestCount | <p data-bbox="829 226 1487 405">Number of input requests that MediaPackage receives. If MediaPackage doesn't receive any requests for input in the specified interval, then no data is given.</p> <p data-bbox="829 449 1011 483">Units: Count</p> <p data-bbox="829 527 1045 560">Valid statistics:</p> <ul data-bbox="829 604 1406 693" style="list-style-type: none"><li>• Sum – Total number of input manifest requests that MediaPackage receives.</li></ul> <p data-bbox="829 764 1084 798">Valid dimensions:</p> <ul data-bbox="829 842 1503 1528" style="list-style-type: none"><li>• ChannelGroup</li><li>• StatusCode</li><li>• Combination of ChannelGroup and StatusCode</li><li>• Combination of ChannelGroup and Channel</li><li>• Combination of ChannelGroup , Channel, and IngestEndpoint</li><li>• Combination of ChannelGroup , Channel, and StatusCode</li><li>• Combination of ChannelGroup , Channel, IngestEndpoint , and StatusCode</li><li>• No dimension</li></ul> |

| Metric              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IngressResponseTime | <p>The time that it takes MediaPackage to process each input request. If MediaPackage doesn't receive any requests for input in the specified interval, then no data is given.</p> <p>Units: Milliseconds</p> <p>Valid statistics:</p> <ul style="list-style-type: none"><li>• <b>Average</b> – Average amount of time (Sum/SampleCount ) that it takes MediaPackage to process input requests over the configured interval.</li><li>• <b>Maximum</b> – Longest amount of time (in milliseconds) that it takes MediaPackage to process an input request and provide a response.</li><li>• <b>Minimum</b> – Shortest amount of time (in milliseconds) that it takes MediaPackage to process an input request and provide a response.</li><li>• <b>SampleCount</b> – Number of requests that's used in the statistical calculation.</li><li>• <b>Sum</b> – Total amount of time that it takes MediaPackage to process input requests over the configured interval.</li></ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"><li>• ChannelGroup</li><li>• Combination of ChannelGroup and Channel</li><li>• Combination of ChannelGroup , Channel, and IngestEndpoint</li></ul> |

| Metric | Description                                                  |
|--------|--------------------------------------------------------------|
|        | <ul style="list-style-type: none"><li>No dimension</li></ul> |

 **Note**

The MultiviewLayout dimension applies to Dynamic Multiview requests only. For a description of the MultiviewLayout and MultiviewSource dimensions, see [MediaPackage live dimensions](#).

## MediaPackage live dimensions

You can filter the AWS/MediaPackage data using the following dimensions.

| Dimension                     | Description                                                                                                                                                                                                                                                                                                                     |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| No Dimension                  | Metrics are aggregated and shown for all channels, endpoints, or status codes.                                                                                                                                                                                                                                                  |
| CDNAuthorizationStatus        | <p>Value: Authorized or Unauthorized</p> <p>Can be used in combination with ChannelGroup , Channel, OriginEndpoint , and CDNAuthorizationStatusDetails to show metrics for the CDN authorization requests to the specified endpoint.</p>                                                                                        |
| CDNAuthorizationStatusDetails | <p>Value when CDNAuthorizationStatus is Authorized : HeaderSecretMatched</p> <p>Value when CDNAuthorizationStatus is Unauthorized : HeaderSecretMismatched , MissingCdnAuthHeaderAndConfiguration , MissingCdnAuthHeader , MissingCdnAuthConfiguration , MissingCdnIdentifierSecretArns , MissingSecretsRoleArn , SecretsMa</p> |

| Dimension | Description                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           | <p>nagerInvalidParameterError , SecretsManagerInternalServiceError , SecretsManagerThrottlingError , MediaPackageSecretsValidationError , or OtherErrors .</p> <p>Can be used in combination with ChannelGroup , Channel, OriginEndpoint , and CDNAuthorizationStatus to show the CDN authorization request details for the specified endpoint.</p>                                                        |
| Channel   | <p>Metrics are shown only for the specified channel.</p> <p>Value: The auto-generated name of the channel.</p> <p>Can be used alone or with other dimensions:</p> <ul style="list-style-type: none"><li>Alone to show metrics for only the specified channel.</li><li>With the originEndpoint dimension to show metrics for the specified endpoint that's associated with the specified channel.</li></ul> |

| Dimension      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChannelGroup   | <p>Metrics are shown only for the specified channel group.</p> <p>Value: The name of the channel group.</p> <p>Can be used alone or with other dimensions:</p> <ul style="list-style-type: none"><li>• Alone to show metrics only for the specified channel group.</li><li>• With the <code>channel</code> dimension to show metrics for the specified channel that's associated with the specified channel group.</li><li>• With the <code>statusCode</code> dimension to show metrics for the specified status code ranges that are associated with the specified channel group.</li></ul> |
| IngestEndpoint | <p>Metrics are shown only for the specified ingest endpoint on a channel.</p> <p>Value: The auto-generated GUID of the ingest endpoint.</p> <p>Can be used with the following dimensions:</p> <ul style="list-style-type: none"><li>• With the <code>channel</code> dimension to show metrics for the specified ingest endpoint that's associated with the specified channel.</li><li>• With the <code>originEndpoint</code> dimension to show metrics for the specified ingest endpoint that's associated with the specified endpoint.</li></ul>                                            |

| Dimension       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MultiviewLayout | <p>The tile layout requested for a Dynamic Multiview stream. Applies to Dynamic Multiview requests only.</p> <p>Value: A layout short name, such as 4E or 3PL.</p> <p>MediaPackage adds this dimension only when a request specifies a valid layout in the <code>aws.multiview</code> query parameter (for example, <code>aws.multiview=layout:4E;sources:cam1,cam2,cam3,cam4</code> ). It isn't emitted for a request that omits <code>aws.multiview</code> , that includes <code>aws.multiview</code> without a <code>layout:</code> value, or that specifies a layout that isn't valid.</p> <p>Added to the <code>EgressRequestCount</code> , <code>EgressBytes</code> , and <code>EgressResponseTime</code> metrics.</p> |
| MultiviewSource | <p>The name of a source channel that is used as a view in a Dynamic Multiview stream. Applies to Dynamic Multiview requests only.</p> <p>Value: The name of a source channel named in the <code>aws.multiview</code> query parameter.</p> <p>Added to the Dynamic Multiview metrics, such as <code>MultiviewSourceRequestCount</code> and <code>MultiviewSourceSegmentMissing</code> . For more information, see <a href="#">Dynamic Multiview metrics</a>.</p>                                                                                                                                                                                                                                                              |

| Dimension      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OriginEndpoint | <p>Metrics are shown for the specified channel and endpoint combination.</p> <p>Value: The auto-generated name of the endpoint.</p> <p>Must be used with the <code>channel</code> dimension.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| RequestType    | <p>Metrics are shown only for the specified request type.</p> <p>Value: Either <code>manifest</code> or <code>segment</code>, signifying the type of content being filtered in the metric.</p> <p>Can be used alone or with other dimensions:</p> <ul style="list-style-type: none"><li>• Alone to show metrics only for the specified request type.</li><li>• With the <code>statusCode</code> dimension to show metrics for the specified request type that's associated with the specified status code range.</li><li>• With the <code>originEndpoint</code> dimension to show metrics for the specified request type that's associated with the specified origin endpoint.</li></ul> |

| Dimension  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StatusCode | <p>Metrics are shown for the specified status code range.</p> <p>Value: 2xx, 3xx, 4xx, or 5xx.</p> <p>Can be used alone or with other dimensions:</p> <ul style="list-style-type: none"><li>• Alone to show all output requests for the specified status range.</li><li>• With the <code>channel</code> dimension to show output requests for all endpoints that are associated with the specified channel, with the specified status code range.</li><li>• With the <code>channel</code> and <code>originEndpoint</code> dimensions to show output requests with a specific status code range on the specified endpoint that's associated with the specified channel.</li></ul> |
| TrackType  | <p>Metrics are shown for the specified track type.</p> <p>Value: Video, Audio, or Subtitle.</p> <p>Can be used alone or with other dimensions:</p> <ul style="list-style-type: none"><li>• Alone to show quality scores for the specified track type.</li><li>• With the <code>channel</code> dimension to show quality scores for all track types that are associated with the specified channel, with the specified track type.</li><li>• With the <code>channel</code> and <code>IngestEndpoint</code> dimensions to show quality scores with a specific track type on the specified ingest endpoint that's associated with the specified channel.</li></ul>                  |

## Dynamic Multiview metrics

When you use Dynamic Multiview, MediaPackage publishes the following additional metrics to the AWS/MediaPackage namespace. These metrics apply to HLS and DASH endpoints. For more information about Dynamic Multiview, see [Dynamic Multiview](#).

MediaPackage also adds the MultiviewLayout dimension to EgressRequestCount, EgressBytes, and EgressResponseTime. The MultiviewLayout and MultiviewSource dimensions are described in [MediaPackage live dimensions](#).

| Metric                      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MultiviewSourceRequestCount | <p>The number of Dynamic Multiview segment requests that included a source channel as a view. Use this metric to see per-source demand. Only segment requests emit this metric; manifest requests don't.</p> <p>Units: Count</p> <p>Valid statistics:</p> <ul style="list-style-type: none"><li>Sum – Total number of source-view requests.</li></ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"><li>Combination of ChannelGroup , Channel, MultiviewSource , and OriginEnd point</li><li>Combination of ChannelGroup , Channel, and MultiviewSource</li><li>Combination of ChannelGroup and MultiviewSource</li><li>ChannelGroup</li><li>No dimension</li></ul> |

| Metric | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | <div><div> <b>Counting behavior</b></div><div>At the dimension levels that don't include <code>MultiviewSource</code> , the value counts view fetches rather than requests: a single multiview segment request contributes one data point for each source it names. These levels are expected to exceed <code>EgressRequestCount</code> for segment requests by roughly the average number of views per request.</div></div> |

| Metric                        | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MultiviewSourceSegmentMissing | <p>The number of times a source channel's segment was missing for a Dynamic Multiview request. MediaPackage replaces the missing view with black video and still serves the response. MediaPackage emits this metric once for each distinct missing source in the request.</p> <p>Units: Count</p> <p>Valid statistics:</p> <ul style="list-style-type: none"><li>• Sum – Total number of missing source segments.</li></ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"><li>• Combination of ChannelGroup , Channel, MultiviewSource , and OriginEndpoint</li><li>• Combination of ChannelGroup , Channel, and MultiviewSource</li><li>• Combination of ChannelGroup and MultiviewSource</li></ul> |

| Metric                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MultiviewSegmentMissing | <p>The aggregate count of missing source segments for Dynamic Multiview requests, without the <code>MultiviewSource</code> dimension. Use this metric to detect whether any view is missing anywhere in a channel group.</p> <p>Units: Count</p> <p>Valid statistics:</p> <ul style="list-style-type: none"><li>Sum – Total number of missing source segments.</li></ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"><li>Combination of <code>ChannelGroup</code> , <code>Channel</code>, and <code>OriginEndpoint</code></li><li>Combination of <code>ChannelGroup</code> and <code>Channel</code></li><li><code>ChannelGroup</code></li></ul> |

| Metric                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MultiviewSegmentIncompatible | <p>The number of Dynamic Multiview segment requests that MediaPackage couldn't assemble because the participating sources' encoder configurations aren't compatible. MediaPackage returns a client error (4xx) for these requests. This metric is aggregate and has no source dimension.</p> <p>Units: Count</p> <p>Valid statistics:</p> <ul style="list-style-type: none"><li>Sum – Total number of segments that couldn't be assembled.</li></ul> <p>Valid dimensions:</p> <ul style="list-style-type: none"><li>Combination of ChannelGroup , Channel, and OriginEndpoint</li><li>Combination of ChannelGroup and Channel</li><li>ChannelGroup</li></ul> |

## Monitoring AWS Elemental MediaPackage with EventBridge events

With Amazon EventBridge, you can automate your AWS services and respond automatically to system events such as application availability issues or error conditions. AWS services deliver events to EventBridge in near real-time. You can write simple rules to indicate which events are of interest to you, and what automated actions to take when an event matches a rule. The actions that can be automatically triggered include the following:

- Invoking an AWS Lambda function
- Invoking AWS Systems Manager Run Command

- Relaying the event to Amazon Kinesis Data Streams
- Activating an AWS Step Functions state machine

For MediaPackage V2, you can use EventBridge and Amazon SNS to be automatically notified when a live-to-VOD harvest job succeeds or fails. MediaPackage emits events on a best-effort basis.

For more information about creating rules in EventBridge, see [Creating rules that react to events in Amazon EventBridge](#) in the *Amazon EventBridge User Guide*.

The following sections describe MediaPackage V2 event types and how to set up notifications of events.

## MediaPackage V2 events

The following topics describe the EventBridge events that MediaPackage V2 emits.

### Harvest job notification events

You get harvest job status events when you export a clip from a live stream to create a live-to-VOD asset. MediaPackage creates notifications when the harvest job succeeds or fails. For information about harvest jobs and live-to-VOD assets, see [Creating live-to-VOD assets with MediaPackage](#).

#### Example Successful harvest job event

```
{
 "id": "8f9b8e72-0b31-e883-f19c-aec84742f3ce",
 "detail-type": "MediaPackageV2 HarvestJob Notification",
 "source": "aws.mediapackagev2",
 "account": "739874020183",
 "time": "2024-11-07T17:32:36Z",
 "region": "us-west-2",
 "resources": [
 "arn:aws:mediapackagev2:us-west-2:111122223333:channelGroup/channel_group_name/channel/channel_name/originEndpoint/origin_endpoint_name/harvestJob/MyHarvestJob"
],
 "detail": {
 "harvestJob": {
 "harvestJobName": "MyHarvestJob",
 "arn": "arn:aws:mediapackagev2:us-west-2:111122223333:channelGroup/channel_group_name/channel/channel_name/originEndpoint/origin_endpoint_name/harvestJob/MyHarvestJob",

```

```

 "status": "COMPLETED",
 "channelGroupName": "channel_group_name",
 "channelName": "channel_name",
 "originEndpointName": "endpoint_name",
 "scheduleConfiguration": {
 "startTime": "2024-11-07T17:29:36Z",
 "endTime": "2024-11-07T17:32:36Z",
 },
 "destination": {
 "s3Destination": {
 "bucketName": "amzn-s3-demo-bucket",
 "destinationPath": "V2Harvests/my-event/"
 },
 },
 "harvestedManifests": {
 "hlsManifests": [
 {
 "manifestName": "examplemanifest.m3u8"
 }
],
 "dashManifests": [
 {
 "manifestName": "examplemanifest2.mpd"
 }
],
 "lowLatencyHlsManifests": [
 {
 "manifestName": "examplemanifest3.m3u8"
 },
 {
 "manifestName": "examplemanifest4.m3u8"
 }
]
 },
 "message": "Your harvest job is complete."
 }
}

```

## Example Failed harvest job event

```

{
 "id": "8f9b8e72-0b31-e883-f19c-aec84742f3ce",

```

```

"detail-type": "MediaPackageV2 HarvestJob Notification",
"source": "aws.mediapackagev2",
"account": "111122223333",
"time": "2024-11-07T17:32:36Z",
"region": "us-west-2",
"resources": [
 "arn:aws:mediapackagev2:us-west-2:111122223333:channelGroup/channel_group_name/
channel/channel_name/originEndpoint/origin_endpoint_name/harvestJob/MyHarvestJob"
],
"detail": {
 "harvestJob": {
 "harvestJobName": "MyHarvestJob",
 "arn": "arn:aws:mediapackagev2:us-west-2:111122223333:channelGroup/
channel_group_name/channel/channel_name/originEndpoint/origin_endpoint_name/harvestJob/
MyHarvestJob",
 "status": "FAILED",
 "channelGroupName": "channel_group_name",
 "channelName": "channel_name",
 "originEndpointName": "endpoint_name",
 "scheduleConfiguration": {
 "startTime": "2024-11-07T17:29:36Z",
 "endTime": "2024-11-07T17:32:36Z",
 },
 "destination": {
 "s3Destination": {
 "bucketName": "amzn-s3-demo-bucket",
 "destinationPath": "V2Harvests/my-event/"
 },
 },
 "harvestedManifests": {
 "hlsManifests": [
 {
 "manifestName": "manifestexample1.m3u8"
 }
],
 "dashManifests": [
 {
 "manifestName": "manifestexample2.mpd"
 }
],
 "lowLatencyHlsManifests": [
 {
 "manifestName": "manifestexample3.m3u8"
 }
],
 },
 },
}

```

```

 {
 "manifestName": "manifestexample4.m3u8"
 }
],
 "message": "There was no content available for the specified time window."
}
}
}

```

## Creating event notifications

You can use Amazon EventBridge and Amazon Simple Notification Service (Amazon SNS) to notify you of new events. In EventBridge, the rule describes which events you're notified about. In Amazon SNS, the topic describes what kind of notification you receive. This section provides high-level steps for creating a topic and rule for events from AWS Elemental MediaPackage. For detailed information about topics and rules, see the following:

- [Create a topic](#) and [Subscribe to a topic](#) in the *Amazon Simple Notification Service Developer Guide*
- [Getting started with Amazon EventBridge](#) in the *Amazon EventBridge User Guide*

### To create notifications of EventBridge events

1. Access [Amazon SNS](#) and create a topic. Give the topic a descriptive name that you will later recognize.
2. Subscribe to the topic that you just created. Choose what kind of notification you want to receive, and where that notification is sent. For example, for email notifications, choose the **Email** protocol and enter the email address to receive notifications for the endpoint.
3. Access [EventBridge](#) and create a rule that uses a **Custom event pattern**. In the pattern preview space, enter the following:

```

{
 "source": [
 "aws.mediapackagev2"
],
 "detail-type": [
 "detail-type from event"
]
}

```

For `detail-type`, enter the value for the `detail-type` field from the event:

`MediaPackageV2 HarvestJob Notification`

4. Add a target to the rule that you just created. Choose **SNS topic**, and then choose the topic that you created in step 1.
5. Configure the details of the rule, and give it a descriptive name. To start using the rule, make sure it's enabled, and then save it.

## Logging AWS Elemental MediaPackage API calls with AWS CloudTrail

Logging is available with only live workflows in MediaPackage.

MediaPackage is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in MediaPackage. CloudTrail captures all API calls for MediaPackage as events. These include calls from the MediaPackage console and code calls to the MediaPackage API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for MediaPackage. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to MediaPackage, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

### MediaPackage information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in MediaPackage, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your account. For more information, see [Viewing events with CloudTrail event history](#).

For an ongoing record of events in your account, including events for MediaPackage, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all AWS Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for creating a trail](#)
- [CloudTrail supported services and integrations](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

All MediaPackage actions are logged by CloudTrail and are documented in the [MediaPackage Live API reference](#). For example, calls to the `CreateChannel`, `CreateOriginEndpoint`, and `RotateIngestEndpointCredentials` operations generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or IAM user credentials
- Whether the request was made with temporary security credentials for a role or federated user
- Whether the request was made by another AWS service

For more information, see the [CloudTrail userIdentity element](#).

## Understanding MediaPackage log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `UpdateChannel` operation:

```
{
 "eventVersion": "1.08",
 "userIdentity": {
 "type": "AssumedRole",
 "principalId": "ABCDEFGHijkl123456789",
 "arn": "arn:aws:sts::444455556666:assumed-role/Admin/testUser",
 "accountId": "444455556666",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
```

```

 "sessionContext": {
 "attributes": {
 "mfaAuthenticated": "false",
 "creationDate": "2023-03-17T00:50:58Z"
 },
 "sessionIssuer": {
 "type": "Role",
 "principalId": "ABCDEFGHijkl123456789",
 "arn": "arn:aws:iam::444455556666:role/Admin",
 "accountId": "444455556666",
 "userName": "Admin"
 }
 },
 "eventTime": "2023-03-17T00:50:59Z",
 "eventSource": "mediapackagev2.amazonaws.com",
 "eventName": "UpdateChannel",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "203.0.113.17",
 "userAgent": "aws-cli/1.18.147 Python/3.6.5 Darwin/17.7.0 botocore/1.10.70",
 "requestParameters": {
 "description": "updated cloudtrail description",
 "id": "cloudtrail-test"
 },
 "responseElements": {
 "description": "updated cloudtrail description",
 "hlsIngest": {
 "ingestEndpoints": [
 {
 "username": "****",
 "url": "https://mediapackagev2.us-west-2.amazonaws.com/in/v2/8d0ca97840d94b18b37ad292c131bcad/8d0ca97840d94b18b37ad292c131bcad/channel",
 "password": "****",
 "id": "8d0ca97840d94b18b37ad292c131bcad"
 },
 {
 "username": "****",
 "url": "https://mediapackagev2.us-west-2.amazonaws.com/in/v2/8d0ca97840d94b18b37ad292c131bcad/9c17f979598543b9be24345d63b3ad30/channel",
 "password": "****",
 "id": "9c17f979598543b9be24345d63b3ad30"
 }
]
 }
 }
 },

```

```
 "id": "cloudtrail-test",
 "arn": "arn:aws:mediapackagev2:us-west-2:444455556666:channelGroup/ChannelGroupName/channel/8d0ca97840d94b18b37ad292c131bcad"
 },
 "requestID": "fc158262-025e-11e9-8360-6bffa705fbb5",
 "eventID": "e9016b49-9a0a-4256-b684-eed9bd9073ab",
 "readOnly": false,
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "444455556666",
 "eventCategory": "Management"
}
```

## Access logging

AWS Elemental MediaPackage v2 provides access logs that capture detailed information about requests sent to your channels. MediaPackage generates two log types:

- Ingress access logs are for requests sent to the channel's input endpoints
- Egress access logs are for requests sent to the channel's endpoints or packaging group's assets

Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze service performance and troubleshoot issues. They can also help you learn about your customer base and understand your MediaPackage bill.

Access logging is an optional feature of MediaPackage that's disabled by default. After you enable access logging, MediaPackage captures the logs and sends them to a destination. Amazon CloudWatch vending log charges apply. For more information, see [CloudWatch pricing](#).

### Topics

- [Permissions](#)
- [Enable access logging](#)
- [Manage and disable access logging](#)
- [Read access logs](#)

## Permissions

MediaPackage uses CloudWatch vended logs to deliver access logging. To deliver access logs, you need permissions to the logging destination that you specify.

To see the required permissions for each logging destination, choose from the following AWS services in the *Amazon CloudWatch Logs User Guide*.

- [Amazon CloudWatch Logs](#)
- [Amazon Simple Storage Service \(Amazon S3\)](#)
- [Amazon Data Firehose](#)

To create, view, or modify logging configuration in MediaPackage, you must have the required permissions. Your IAM role must include the following minimum permissions to manage access logging in the MediaPackage console.

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "ServiceLevelAccessForLogDelivery",
 "Effect": "Allow",
 "Action": [
 "logs:CreateLogGroup",
 "logs:CreateLogStream",
 "logs:PutLogEvents"
],
 "Resource": "arn:aws:mediapackagev2:us-
east-1:123456789012:channelGroup/*"
 }
]
}
```

For more information about permissions to manage access logging, see [Enable logging from AWS services](#) in the *Amazon CloudWatch Logs User Guide*.

## Enable access logging

After you set up permissions to the logging destination, you can enable access logging for MediaPackage.

For each log type, you can configure up to 3 log deliveries.

### To enable access logs for an existing channel (console)

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.
2. Select your channel group from the list of channel groups.
3. Under **Access Logging**, do the following:
  - a. For **Log deliveries – Egress Access Logs** or **Log deliveries – Ingress Access Logs**, choose **Add**, and then do the following:
  - b. Choose one of the following logging destinations.
    - Amazon CloudWatch Logs
    - Amazon S3
    - Firehose

#### Tip

- If you choose Amazon S3 or Firehose, you can deliver your logs to a **Cross account** or **In current account**.
- To enable cross-account delivery, both AWS accounts must have the required permissions. For more information, see the [Cross-account delivery example](#) in the *Amazon CloudWatch Logs User Guide*.

4. For the **Delivery destination ARN**, choose or enter the ARN. If you don't have one already, follow the prompts to create one.
5. For **Additional settings - optional**, choose the following:
  - a. For **Field selection**, select the log fields to include in each log record.
  - b. For **Output format**, choose the output format for the log.
  - c. For **Field delimiter**, choose how to separate each log field.

- d. (Amazon S3) For **Suffix**, specify the suffix path to partition your data. You can use the following fields: {accountid}, {region}, {channel\_group\_id}, {yyyy}, {MM}, {dd}, {HH}
  - e. (Amazon S3) For **Hive-compatible**, choose **Enable** if you want to use Hive-compatible S3 paths.
6. To apply your changes to all log types, choose **Apply to all log types**.
  7. To create another log destination, repeat steps 3 – 5.
  8. Follow the prompts to update your channel group.

You can also use the CloudWatch API to enable access logs for your channel groups.

### To enable access logs for an existing channel (API)

1. After you create a channel group by using either the MediaPackage v2 API or the MediaPackage v2 console, get the Amazon Resource Name (ARN) of the channel group.

You can find the ARN from the **Channel groups** page in the MediaPackage console or you can use the [GetChannel](#) API operation. A channel group ARN follows this format:

arn:aws:mediapackagev2:*region*:*123456789012*:*channelGroup*/*channelGroupName*

2. Next, use the CloudWatch [PutDeliverySource](#) API operation to create a delivery source for the channel group.
  - a. Pass the `resourceArn`.
  - b. For `logType`, specify the type of logs that are collected:
    - `EGRESS_ACCESS_LOGS` for requests sent to the endpoint or asset
    - `INGRESS_ACCESS_LOGS` for requests sent to channel

### Example PutDeliverySource operation

```
{
 "logType": "EGRESS_ACCESS_LOGS",
 "name": "my-channel-group-source",
 "resourceArn":
 "arn:aws:mediapackagev2:region:123456789012:channelGroup/channelGroupName"
}
```

3. Use the [PutDeliveryDestination](#) API operation to configure where to store your logs. You can choose either CloudWatch Logs, Amazon S3, or Firehose as the destination. You must specify the ARN of one of the destination options for where your logs will be stored. You can choose the `outputFormat` of the logs to be one of the following: `json`, `plain`, `w3c`, `raw`, `parquet`.

The following is an example of configuring logs to store in an Amazon S3 bucket and in JSON format.

```
{
 "deliveryDestinationConfiguration": {
 "destinationResourceArn": "arn:aws:s3:::amzn-s3-demo-bucket-name"
 },
 "name": "string",
 "outputFormat": "json",
 "tags": {
 "key" : "value"
 }
}
```

#### Note

If you're delivering logs cross-account, you must use the `PutDeliveryDestinationPolicy` API to assign an AWS Identity and Access Management (IAM) policy to the destination account. The IAM policy allows delivery from one account to another account.

4. Use the [CreateDelivery](#) API operation to link the delivery source to the destination that you created in the previous steps. This API operation associates the delivery source with the end destination.

```
{
 "deliveryDestinationArn": "string",
 "deliverySourceName": "string",
 "tags": {
 "string" : "string"
 }
}
```

## Manage and disable access logging

Follow these procedures to manage or disable access logging for MediaPackage.

### To manage access logging for a channel (console)

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.
2. Select your channel group from the list of channel groups.
3. In the **Access Logging** section, select the destination and choose **Edit**.
4. Modify the log configuration as needed and then choose **Update**.

#### Note

You can't modify the logging destination. If you need to change the logging destination, delete the current configuration and create a new logging destination.

You can disable access logs for your MediaPackage channel at any time.

### To disable access logging for a channel (console)

1. Open the MediaPackage console at <https://console.aws.amazon.com/mediapackage/>.
2. Select your channel group from the list of channel groups.
3. In the **Access Logging** section, select the destination to disable and choose **Remove**.
4. Review your changes and then choose **Delete**.

## Read access logs

MediaPackage v2 uses ingestion hub to deliver your access logs.

If you're using CloudWatch Logs as the destination, you can use CloudWatch Logs Insights to read the access logs. Typical CloudWatch Logs charges apply. For more information, see [Analyzing Log Data with CloudWatch Logs Insights](#) in the *AWS CloudWatch Logs User Guide*.

**Note**

Access logs can take a few minutes to appear in your destination. If you don't see the logs, wait a few minutes and try again.

**Access log format**

The access log files consist of a sequence of formatted log records, where each log record represents one request. The order of the fields within the log can vary.

**Example: Ingress access log**

```
{
 "event_timestamp": "2020-07-13T18:59:56.293656Z",
 "resource_arn": "arn:aws:mediapackagev2:us-east-1:007862077394:123456789012/my_channel_group",
 "client_ip": "192.0.2.0/24",
 "time_to_first_byte": 0.445,
 "status_code": "200",
 "received_bytes": 468,
 "sent_bytes": 2587370,
 "method": "GET",
 "request": "https://aabbcc-1.ingest.eeffgg.mediapackagev2.us-east-1.amazonaws.com:443/in/v1/my_channel_group/1/my_channel/manifest.m3u8",
 "protocol": "HTTP/1.1",
 "user_agent": "sabr/3.0 Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Safari/528.17",
 "account": "123456789012",
 "channel_id": "my_channel",
 "channel_arn": "arn:aws:mediapackage:us-west-2:111122223333:channels/ExampleChannelID",
 "domain_name": "aaabbbcccdddee.mediapackage.us-east-1.amazonaws.com",
 "request_id": "aaaAAA111bbbBBB222cccCCC333dddDDD",
 "channel_group_id": "my_channel_group",
 "input_type": "HLS",
 "input_index": "1"
}
```

**Example: Egress access log**

```
{
```

```

 "event_timestamp": "2020-07-13T18:59:56.293656Z",
 "resource_arn": "arn:aws:mediapackagev2:us-east-1:007862077394:123456789012/my_channel_group",
 "client_ip": "192.0.2.0/24",
 "time_to_first_byte": 0.445,
 "status_code": "200",
 "received_bytes": 468,
 "sent_bytes": 2587370,
 "method": "GET",
 "request": "https://aabbcc.egress.eeffgg.mediapackagev2.us-east-1.amazonaws.com:443/out/v1/my_channel_group/my_channel/my_endpoint/index.mpd?param1=value1¶m2=value2",
 "request_query_params": "param1=value1¶m2=value2",
 "protocol": "HTTP/1.1",
 "user_agent": "sabr/3.0 Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Safari/528.17",
 "account": "111122223333",
 "channel_id": "my_channel",
 "channel_arn": "arn:aws:mediapackage:us-west-2:123456789012:channels/ExampleChannelID",
 "domain_name": "aaabbbcccdddee.mediapackage.us-east-1.amazonaws.com",
 "request_id": "aaaAAA111bbbBBB222cccCCC333dddDDD",
 "endpoint_id": "my_endpoint",
 "endpoint_arn": "arn:aws:mediapackage:us-west-2:123456789012:origin_endpoints/ExampleEndpointID",
 "channel_group_id": "my_channel_group",
 "manifest_name": "index",
 "manifest_type": "DASH"
}

```

The following list describes the log record fields, in order:

### **event\_timestamp**

The time of day when the request was received. The value is ISO-8601 date time and is based on the system clock of the host that served the request.

### **resource\_arn**

The Amazon Resource Name (ARN) of the channel group that received the request.

### **client\_ip**

The IP address of the requesting client.

**time\_to\_first\_byte**

The number of seconds that MediaPackage spent processing your request. This value is measured from the time the last byte of your request was received until the time the first byte of the response was sent.

**status\_code**

The numeric HTTP status code of the response.

**received\_bytes**

The number of bytes in the request body that the MediaPackage server receives.

**sent\_bytes**

The number of bytes in the response body that the MediaPackage server sends. This value often is the same as the value of the Content-Length header that's included with server responses.

**method**

The HTTP request method that was used for the request: DELETE, GET, HEAD, OPTIONS, PATCH, POST, or PUT.

**request**

The request URL.

**protocol**

The type of protocol used for the request, such as HTTP.

**user\_agent**

A user-agent string that identifies the client that originated the request, enclosed in double quotes. The string consists of one or more product identifiers, product/version. If the string is longer than 8 KB, it is truncated.

**account**

The AWS account ID of the account that was used to make the request.

**channel\_id**

The ID of the channel that received the request.

**channel\_arn**

The Amazon Resource Name (ARN) of the channel that received the request.

**domain\_name**

The server name indication domain provided by the client during the TLS handshake, enclosed in double quotes. This value is set to - if the client doesn't support SNI or the domain doesn't match a certificate and the default certificate is presented to the client.

**request\_id**

A string that's generated by MediaPackage to uniquely identify each request.

**endpoint\_id**

The ID of the endpoint that received the request.

**endpoint\_arn**

The Amazon Resource Name (ARN) of the endpoint that received the request.

**input\_type**

The ingest file formats and protocol.

**input\_index**

The input source index.

**manifest\_name**

The name of the egress request manifest request. Remains empty for segment egress request.

**manifest\_type**

The type of the egress request manifest request. Remains empty for segment egress request.

**request\_query\_params**

The manifest filtering query parameter names and values.

## Examples

The following are queries that you can use to read MediaPackage debug log data.

**Example View the HTTP status code responses for a channel**

Use this query to view the responses by HTTP status code for a channel. You can use this to view HTTP error code responses to help you to troubleshoot issues.

```
fields @timestamp, @message
```

```
| filter channelId like my-channel
| stats count() by statusCode
```

### Example Get the number of requests per endpoint on a channel

```
fields @timestamp, @message
| filter channelId like my-channel
| stats count() by endpointId
```

## Monitoring manifest update time in AWS Elemental MediaPackage

MediaPackage playback responses include the following custom headers that indicate when MediaPackage last modified the manifest in non-dynamic ad insertion workflows. These headers are helpful when troubleshooting issues related to stale manifests.

### X-Amzn-Mediapackage-Manifest-Last-Part

This is only provided on requests for low-latency HLS manifests, and is the highest part sequence number in the manifest.

### X-MediaPackage-Manifest-Last-Sequence

This is the highest segment sequence number in the manifest.

For HLS and CMAF, this is the highest segment number in the media playlist.

See the following section for [manifest example](#).

### X-MediaPackage-Manifest-Last-Updated

The epoch timestamp in milliseconds when MediaPackage generates the segment referred to in X-MediaPackage-Manifest-Last-Sequence.

## HLS manifest example

### HLS manifest

MediaPackage determines the X-MediaPackage-Manifest-Last-Sequence value from the last segment in the manifest. For example, in the following manifest `index_1_3.ts` is the highest

segment sequence number, so the value of X-MediaPackage-Manifest-Last-Sequence is 3. The value of X-MediaPackage-Manifest-Last-Updated corresponds to the epoch timestamp in milliseconds when MediaPackage generates the last segment in the manifest.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:8
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:7.500,
index_1_0.ts?m=1583172400
#EXTINF:7.500,
index_1_1.ts?m=1583172400
#EXTINF:7.500,
index_1_2.ts?m=1583172400
#EXTINF:7.500,
index_1_3.ts?m=1583172400
#EXT-X-ENDLIST
```

## MediaPackage response headers

Use the following AWS Elemental MediaPackage response headers to help you build your workflows. For more information about response headers associated with manifest monitoring, see [Monitoring manifest update time](#).

| Header name                      | Value                                                                                            | Description                                                                                                |
|----------------------------------|--------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| CMSD-Static                      | For accepted key/value pairs, see <a href="#">CMSD headers from AWS Elemental MediaPackage</a> . | A variety of Common Media Server Data (CMSD) information about the manifests and segment response objects. |
| X-Amzn-Mediapackage-Active-Input | 1 or 2                                                                                           | The active input pipeline when MediaPackage serves a given media segment.                                  |
| X-Amzn-Mediapackage-Channel-Id   | The channel-name value in the API.                                                               | Use separately from, or in addition to, X-Amzn-MediaPackage-Channel-UniqueId , to identify                 |

| Header name                          | Value                                 | Description                                                                                                                                                                                                                                                 |
|--------------------------------------|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                      |                                       | a given channel in the CDN logs. Channel names are unique only within a given region.                                                                                                                                                                       |
| X-Amzn-Mediapackage-Channel-UniqueId | The unique identifier of the channel. | Use separately from, or in addition to, X-Amzn-MediaPackage-Channel-Id , to identify a given channel in the CDN logs. Channel names are unique only within a given region. Using X-Amzn-Mediapackage-Channel-UniqueId is also helpful for support requests. |
| X-Amzn-Mediapackage-Endpoint-Id      | The manifest-name value in the API.   | Use separately from, or in addition to, X-Amzn-Mediapackage-Endpoint-UniqueId , to identify a given endpoint in the CDN logs. Endpoint names are unique only within a given channel and region.                                                             |

| Header name                           | Value                                  | Description                                                                                                                                                                                                                                                                 |
|---------------------------------------|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| X-Amzn-Mediapackage-Endpoint-UniqueId | The unique identifier of the endpoint. | Use separately from, or in addition to, X-Amzn-MediaPackage-Endpoint-Id , to identify a given endpoint in the CDN logs. Endpoint names are unique only within a given channel and region. Using X-Amzn-Mediapackage-Endpoint-UniqueId is also helpful for support requests. |
| X-Amzn-RequestId                      | The unique identifier of the request.  | Equivalent to X-Amzn-MediaPackage-Request-Id in MediaPackage V1. Using X-Amzn-RequestId is helpful for support requests.                                                                                                                                                    |

# Tagging AWS Elemental MediaPackage resources

A tag is a label that you assign to an AWS resource. Each tag consists of a *key* and a *value*, both of which you define. For example, the key might be "stage" and the value might be "test". You can use tags for a variety of purposes. One common use is to control access to AWS resources using tags. For information, see the [Controlling access to AWS resources using tags](#) topic in the *IAM User Guide*.

Another common use of tags is to categorize and track your MediaPackage costs. When you apply cost allocation tags to MediaPackage channels, endpoints, and packaging configurations, AWS generates a cost allocation report as a comma-separated value (CSV) file with your usage and costs aggregated by your tags. You can apply tags that represent business categories (such as cost centers, application names, or owners) to organize your costs across multiple services. For more information about using tags for cost allocation, see [Using cost allocation tags](#) in the [AWS Billing User Guide](#).

## Tag restrictions

The following restrictions apply to tagging AWS Elemental MediaPackage resources:

- Cost allocation tagging is only available for channel, endpoint, and packaging configuration resources. You can't use cost allocation tags for asset or packaging group resources.
- Maximum number of tags that you can assign to a resource – 50.
- Maximum key length – 128 Unicode characters.
- Maximum value length – 256 Unicode characters.
- Valid characters for key and value – a-z, A-Z, 0-9, space, and the following characters: \_ . : / = + - and @.
- Keys and values are case sensitive.
- Don't use `aws:` as a prefix for keys; it's reserved for AWS use.
- Can't be used for harvested live-to-VOD assets.

## Managing tags

You can use the AWS Elemental MediaPackage API or the AWS CLI to add, edit, or delete the values for these properties.

For more information, see the actions related to tags in the following reference documentation:

- [Tags resource-arn](#) in the *AWS Elemental MediaPackage live API reference*.
- [tag-resource](#) in the *AWS CLI MediaPackage reference*.

# Quotas in AWS Elemental MediaPackage

This section describes the quotas for AWS Elemental MediaPackage. For information about requesting an increase to soft quotas, see [AWS service quotas](#). You can't request an increase to hard quotas.

## Topics

- [Soft quotas](#)
- [Hard quotas](#)

## Soft quotas

The following table describes quotas in AWS Elemental MediaPackage that can be increased. For information about changing quotas, see [AWS Service Quotas](#).

For some customers, your account quota might be below these published quotas. If you believe that you encountered a Resource limit exceeded error wrongfully, use the Service Quotas console to [request quota increases](#).

| Resource or operation         | Default quota                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Maximum active harvest jobs   | 10 per channel group                                                                                                                                                                                                                                                                                                                                                            |
| Maximum channel groups        | 3 per account                                                                                                                                                                                                                                                                                                                                                                   |
| Maximum channels              | 10 per channel group                                                                                                                                                                                                                                                                                                                                                            |
| Maximum endpoints per channel | 10 per channel<br><br>Each origin endpoint represents the output package that you use. If one channel serves TS or TS encrypted, CMAF or CMAF encrypted content, then that channel has 4 endpoints and falls within the 10 endpoints quota. If you have 10 channels set up this same way, then you still haven't exceeded the quota because each channel uses only 4 endpoints. |

| Resource or operation                 | Default quota                    |
|---------------------------------------|----------------------------------|
| Maximum live manifest length          | 15 minutes                       |
| Maximum manifests per origin endpoint | 25 manifests per origin endpoint |

## Hard quotas

The following table describes quotas in AWS Elemental MediaPackage that can't be increased.

| Resource or operation                        | Quota                                                                                                                                               |
|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Maximum content age for time-shifted viewing | 336 hours (14 days)                                                                                                                                 |
| Maximum request rates per channel input      | 200 requests per second                                                                                                                             |
| Maximum request rates per endpoint           | <ul style="list-style-type: none"><li>Media segments output: 500 requests per second</li><li>Manifests output: 10,000 requests per second</li></ul> |

 **Note**

The per endpoint origination request rate quotas are indicative only and based on typical traffic patterns when using a properly configured CDN. The request rate quotas are applicable for live events, linear channels, and time-shifted viewing. The request rate quotas may be lower under certain conditions like misconfigured CDNs or players generating abnormal levels of origin requests with unique HTTP

| Resource or operation                | Quota                                                                                                                        |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
|                                      | header values, or unique query string values appended to the playback URLs.                                                  |
| Maximum REST API requests            | <ul style="list-style-type: none"><li>Steady state: 5 requests per second</li><li>Bursting: 50 requests per second</li></ul> |
| Maximum time-shifted manifest length | 24 hours for all supported output formats                                                                                    |
| Maximum tracks per ingest stream     | 10<br><br>The maximum number of tracks (audio, video, subtitle, etc.) per stream that you can ingest.                        |

# Document history for the MediaPackage User Guide

The following table describes the documentation releases for MediaPackage.

| Change                                                           | Description                                                                                                                                  | Date              |
|------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| <a href="#">Added output locking mode documentation</a>          | MediaPackage now supports non-epoch-locked mode for CMAF channels.                                                                           | July 23, 2026     |
| <a href="#">Added DASH audio timeline pattern information</a>    | Added the topic that describes the audio timeline pattern option for DASH manifests that MediaPackage serves.                                | June 1, 2026      |
| <a href="#">Added content key encryption for MediaPackage V2</a> | Added support for content key encryption in MediaPackage V2 using certificates from AWS Certificate Manager.                                 | November 17, 2025 |
| <a href="#">Added CMAF ingest documentation</a>                  | MediaPackage now supports enhanced features for stream control when using MediaPackage output groups in AWS Elemental MediaLive.             | October 1, 2025   |
| <a href="#">Added support for HLS CUE tags</a>                   | MediaPackage now supports HLS and LL-HLS CUE tag ad markers for <code>time_signal</code> or <code>splice_insert</code> tags in the manifest. | September 9, 2025 |
| <a href="#">Added support for preferred input selection</a>      | You can now choose which input is the preferred default when both inputs have the same health.                                               | July 25, 2025     |

|                                                              |                                                                                                                                                          |                |
|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| <a href="#">Added CDN static auth headers info</a>           | Added the topic that describes how to set up CDN authorization if your CDN doesn't support AWS SigV4.                                                    | July 16, 2025  |
| <a href="#">Added managing DRM segment metadata</a>          | Added the topic that describes how to exclude SEIG and SGPD metadata boxes from CMAF endpoints with encryption.                                          | July 3, 2025   |
| <a href="#">Added Microsoft Smooth Streaming information</a> | Added the topic that describes how MSS endpoints work in MediaPackage.                                                                                   | July 3, 2025   |
| <a href="#">Support for DVB-DASH and EBU-TT-D subtitles</a>  | Added descriptions for new fields in support of added DVB-DASH functionality.                                                                            | May 20, 2025   |
| <a href="#">Added DASH compactness information</a>           | Added the topic that describes options for compacting or expanding DASH manifests that MediaPackage serves.                                              | April 14, 2025 |
| <a href="#">Added EventBridge events information</a>         | Added the topic that describes how to use Amazon EventBridge and Amazon Simple Notification Service to notify you of success or failure of harvest jobs. | March 26, 2025 |
| <a href="#">URL encoding query parameters</a>                | Added information about how to URL encode the query parameters in manifest requests to MediaTailor.                                                      | March 12, 2025 |

|                                                                         |                                                                                                                                            |                   |
|-------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| <a href="#">Added reset information</a>                                 | Added new topic that describes use cases for history reset for channels and origin endpoints.                                              | March 12, 2025    |
| <a href="#">Added HarvestObject content</a>                             | Added information the HarvestObject dataplane API operation.                                                                               | January 28, 2025  |
| <a href="#">Added formatting for query parameters</a>                   | Added information about how manifest filtering query parameters can be formatted to include multiple queries.                              | January 16, 2025  |
| <a href="#">Added content about limiting the duration of a manifest</a> | Added information about manifest_window_seconds query parameters, which can be used to request shorter manifests on a session-level basis. | January 16, 2025  |
| <a href="#">Added clip-start time delay</a>                             | Added information about using clip-start query parameters with time delayed live content.                                                  | January 16, 2025  |
| <a href="#">Added support for MQCS</a>                                  | MediaPackage now supports Media Quality Confidence Scores (MQCS), which indicates the quality of a stream down to the segment.             | November 21, 2024 |
| <a href="#">Added support for CMSD headers</a>                          | MediaPackage now supports Common Media Server Data (CMSD) headers in responses to downstream systems.                                      | November 21, 2024 |

|                                                           |                                                                                                                                       |                   |
|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| <a href="#">Added support for access logging</a>          | MediaPackage v2 supports access logging and sending your logs to the log destination that you specify.                                | November 15, 2024 |
| <a href="#">Harvest jobs</a>                              | Harvest jobs for creating live-to-VOD assets are supported in MediaPackage v2.                                                        | October 30, 2024  |
| <a href="#">Added support for Dolby Vision 8.1</a>        | MediaPackage now supports the video codec Dolby Vision 8.1, which increases the range of colors that can be shown in a video.         | July 23, 2024     |
| <a href="#">Added cross-region failover content</a>       | Added information about cross-region failover in AWS Elemental MediaPackage                                                           | June 11, 2024     |
| <a href="#">Live manifest length quota</a>                | Moved Maximum Live Manifest Length to Live soft quotas table. This change allows you to submit requests to increase your quota limit. | February 14, 2024 |
| <a href="#">Manifest header</a>                           | MediaPackage added manifest header for low-latency HLS manifests.                                                                     | January 31, 2024  |
| <a href="#">Manifest filters and filter configuration</a> | Updated the note for clarification when configuring Manifest filters as part of the Filter Configuration.                             | January 31, 2024  |
| <a href="#">Manifest filters</a>                          | MediaPackage added support for manifest filters for endpoint egress requests.                                                         | October 30, 2023  |

|                                                                                     |                                                                                                                                       |                  |
|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">Manifest filters</a>                                                    | MediaPackage added support for manifest filters for endpoint egress requests.                                                         | October 30, 2023 |
| <a href="#">Response headers</a>                                                    | Added section that explains MediaPackage response headers.                                                                            | August 4, 2023   |
| <a href="#">AWS managed policy updates</a>                                          | Added new AWS managed policies: <code>AWSElementalMediaPackageV2FullAccess</code> , <code>AWSElementalMediaPackageV2ReadOnly</code> . | July 25, 2023    |
| <a href="#">time_delay_query parameters added to time-shifted viewing reference</a> | Added information on using <code>time_delay_query</code> parameters.                                                                  | July 23, 2023    |
| <a href="#">Added data plane APIs</a>                                               | Added <code>PutObject</code> , <code>GetObject</code> , and <code>GetHeadObject</code> data plane APIs.                               | June 14, 2023    |
| <a href="#">Initial release</a>                                                     | Initial release of the AWS Elemental MediaPackage User Guide                                                                          | May 5, 2023      |